# MULTIPLICADOR EN EL CUERPO FINITO GF($2^{163}$) USANDO BASES NORMALES GAUSSIANAS

*Vladimir Trujillo, Jaime Velasco-Medina, Julio C. López-Hernández*

Grupo de Bioelectrónica y Nanoelectrónica, EIEE, Universidad del Valle
A.A. 25360, Cali, Colombia
E-mail: vlatruo@yubarta.univalle.edu.co, jvelasco@univalle.edu.co, jlopez@eisc.univalle.edu.co

## ABSTRACT

In this article we present an architecture for a multiplier circuit in the field GF($2^{163}$) using gaussian normal basis. This multiplier is the main functional block of the arithmetic unit of a cryptographic processor for elliptic curves. The hardware architecture is described using VHDL and synthesized by MAX+plus II on the CPLD EPF10K200SFC672-1X. The simulation results show a very good performance for this multiplier.

## RESUMEN

En este articulo se presenta una arquitectura para un multiplicador en el cuerpo finito binario GF($2^{163}$) usando bases normales gaussianas. Este multiplicador constituye el principal bloque funcional de la unidad aritmética de un criptoprocesador para curvas elípticas sobre cuerpos binarios. La arquitectura del hardware es descrita usando VHDL y es sintetizada usando MAX+plus II de Altera sobre el circuito CPLD EPF10K200SFC672-1X. Los resultados de simulación muestran un buen desempeño para este multiplicador.

# MULTIPLICADOR EN EL CUERPO FINITO GF($2^{163}$) USANDO BASES NORMALES GAUSSIANAS

*Vladimir Trujillo, Jaime Velasco-Medina, Julio C. López-Hernández*

Grupo de Bioelectrónica y Nanoelectrónica, EIEE, Universidad del Valle
A.A. 25360, Cali, Colombia
E-mail: vlatruo@yubarta.univalle.edu.co, jvelasco@univalle.edu.co, jlopez@eisc.univalle.edu.co

**RESUMEN**

*En este articulo se presenta una arquitectura para un multiplicador en el cuerpo finito binario GF($2^{163}$) usando bases normales gaussianas. Este multiplicador constituye el principal bloque funcional de la unidad aritmética de un criptoprocesador para curvas elípticas sobre cuerpos binarios. La arquitectura del hardware es descrita usando VHDL y es sintetizada usando MAX+plus II de Altera sobre el circuito CPLD EPF10K200SFC672-1X. Los resultados de simulación muestran un buen desempeño para este multiplicador*

## 1. INTRODUCCION

En los sistemas de comunicación modernos es de vital importancia la confiabilidad y la seguridad de la información. Entonces, en muchos casos con el propósito de alcanzar los requerimientos anteriores se usa la aritmética de cuerpo finito GF($2^m$) para la implementación de algoritmos criptográficos tanto a nivel de software como a nivel de hardware. Sin embargo, debido a la necesidad de cumplir con los requerimientos de alta velocidad de los sistemas criptográficos, la mayoría de las aplicaciones requieren implementaciones a nivel de hardware. En este caso, para el cálculo de las operaciones aritméticas en el cuerpo finito GF($2^m$), se requieren circuitos multiplicadores de alto desempeño, puesto que la mayoría de las funciones aritméticas importantes tales como la inversión y la exponenciación están basadas en la multiplicación.

Diferentes algoritmos para la multiplicación en el cuerpo finito GF($2^m$) tanto a nivel de software como a nivel de hardware (circuitos VLSI o circuitos FPGAs) han sido presentados en la literatura [1][2][3][4][5]. Adicionalmente, se han utilizado diferentes bases para la representación de los elementos del cuerpo finito GF($2^m$) en los que operen tales multiplicadores.

Los criptosistemas de curvas elípticas definidos en los cuerpos binarios GF($2^m$), recomendados por NIST (National Institute of Standards and Technology), ofrecen alta seguridad con claves de tamaño menor que otros criptosistemas de clave pública. Así, por ejemplo, un criptosistema de curvas elípticas definido en el cuerpo binario GF($2^{163}$) requiere claves de 163 bits mientras que el sistema RSA, con una seguridad computacional equivalente, requiere claves de 1024 bits. Entonces, nuestro objetivo es implementar a nivel de hardware algoritmos de multiplicación para cuerpos finitos específicos recomendados por NIST.

Generalmente, el desempeño de un multiplicador se califica con base a dos características principales: el tiempo requerido para realizar la operación, y el número de compuertas lógicas requeridas para su implementación. En este orden de ideas, es importante notar que las arquitecturas paralelas son mas rápidas que las arquitecturas seriales, sin embargo éstas últimas requieren de menos compuertas lógicas para su implementación. Entonces, existe un compromiso entre ambas características que se debe considerar en el momento del diseño. Por ejemplo, para ciertos algoritmos criptográficos donde el tamaño de las claves es de 512 bits ó 1024 bits no es práctico el uso de una arquitectura paralela para el multiplicador, esto exige una gran cantidad de compuertas lógicas, lo que estará limitado por la capacidad de los dispositivos físicos que existen en el mercado. Sin embargo, los algoritmos criptográficos para curvas elípticas donde el tamaño de la clave es de 163 bits ó 255 bits, una multiplicador paralelo es bastante competitivo.

En este trabajo se presenta el diseño de una implementación en hardware de un algoritmo presentado en [6][7], para realizar la multiplicación en el cuerpo finito GF($2^{163}$) usando bases normales gaussianas. Este multiplicador es implementado a partir de algunas modificaciones del algoritmo anterior. La arquitectura del multiplicador es semi-sistólica, debido a que es posible

realizar una descripción VHDL de tipo genérico para la mayoría de los bloques funcionales, es decir, que con un pequeño cambio se puede diseñar un multiplicador para diferentes tamaños de la palabra de los datos de entrada (*m*). De otro lado, debido a que el multiplicador presenta una operación serial, la arquitectura de este utiliza poca área, entonces es posible realizar una arquitectura paralela para procesar varios datos simultáneamente, lo cual permite incrementar la velocidad de operación.

Este articulo esta organizado de la siguiente manera, la sección 2 presenta una descripción de las operaciones aritméticas para bases normales gaussianas en el cuerpo finito GF($2^m$), la sección 3 describe el algoritmo de multiplicación a nivel de bits para bases normales gaussianas en el cuerpo finito GF($2^{163}$), la sección 4 presenta el diseño de la nueva arquitectura para el multiplicador en el cuerpo finito GF($2^{163}$) usando bases normales gaussianas, la sección 5 muestra los resultados de las simulaciones. Finalmente, la sección 6 presenta algunas conclusiones y el trabajo futuro.

## 2. OPERACIONES ARITMÉTICAS PARA BASES NORMALES GAUSSIANAS EN GF($2^m$)

### 2.1 Conceptos sobre bases normales gaussianas

La representación de un elemento en el cuerpo finito GF($2^{163}$) usando bases normales presenta una ventaja computacional debido a que permite realizar el calculo del cuadrado de un elemento de una manera eficiente. Sin embargo, multiplicar elementos distintos es algo complicado. Por lo tanto, con el propósito de solucionar esta dificultad, ANSI X9.62 recomienda que las *Bases Normales Gaussianas* (BNG) sean usadas, las cuales permiten realizar la multiplicación de una manera más simple y eficiente. Los conceptos básicos sobre bases normales gaussianas son presentados en [8].

Una base normal para GF($2^m$) es de la forma:

$\{\beta, \beta^2, \beta^{2^2}, ..., \beta^{2^{m-1}}\}$, donde $\beta \in$ GF($2^m$),

en donde, cualquier elemento $\alpha \in$ GF($2^m$) puede ser escrito de la forma:

$$\alpha = \sum_{i=0}^{m-1} a_i \beta^{2^i} \text{ donde } a_i \in \{0,1\}.$$

El *tipo T* de una BNG es un entero positivo, el cual mide la complejidad de la multiplicación con respecto a la base. Generalmente, el *tipo T* de menor valor permite realizar una multiplicación mas eficiente. Se puede demostrar que para un *m* y *T* dado, el cuerpo finito GF($2^m$) puede tener al menos una BNG. Entonces, es adecuado hablar del *tipo T* de una BNG en el cuerpo finito GF($2^m$).

Una BNG existe cuando *m* no es divisible por 8. Sea *m* un entero positivo no divisible por 8, y *T* un entero positivo, entonces un *tipo T* de una BNG en el cuerpo finito GF($2^m$) existe si y solo si $p = Tm + 1$ es primo.

Si $\{\beta, \beta^2, \beta^{2^2}, ..., \beta^{2^{m-1}}\}$ es una Base Normal Gaussiana en el cuerpo finito GF($2^m$), entonces el elemento $\alpha = \sum_{i=0}^{m-1} a_i \beta^{2^i}$ es representado por la cadena binaria $(a_0 a_1 a_2 ..... a_{m-1})$ donde $a_i \in \{0,1\}$

En este caso, la identidad multiplicativa es representada por una cadena de unos, mientras que la identidad aditiva es representada por una cadena de ceros.

Un resultado importante para la aritmética de las bases normales gaussianas es el teorema de Fermat. Para todo $\beta \in$ GF($2^m$) se tiene:

$$\beta^{2^m} = \beta$$

Este teorema es importante para realizar el cuadrado de un elemento en el cuerpo finito GF($2^m$).

### 2.2 Operaciones aritméticas para bases normales gaussianas en GF($2^m$)

Las siguientes operaciones aritméticas se definen sobre los elementos de un cuerpo finito GF($2^m$) cuando se usa una BNG de *tipo T* :

- *Adición*:

si $a = (a_0 a_1 a_2 ..... a_{m-1})$ y $b = (b_0 b_1 b_2 ... b_{m-1})$ son elementos de GF($2^m$), entonces
$a + b = c = (c_0 c_1 c_2 ... c_{m-1})$
donde $c_i = (a_i + b_i) \bmod 2$.

- *Cuadrado* :

sea $a = (a_0 a_1 a_2 ..... a_{m-1}) \in$ GF($2^m$), entonces

$$\alpha^2 = \left( \sum_{i=0}^{m-1} \alpha_i \beta^{2^i} \right)^2$$

$= \sum_{i=0}^{m-1} \alpha_i \beta^{2^{i+1}} = \sum_{i=0}^{m-1} \alpha_{i-1} \beta^{2^i}$ debido al teorema de Fermat; $\beta^{2^m} = \beta$

$= a^2 = (a_{m-1} a_0 a_1 a_2 ..... a_{m-2})$

En este caso, el cuadrado es una simple rotación a la derecha de la representación del vector.

- *Multiplicación* :

sea $p = Tm + 1$ y sea $u \in$ GF($p$) un elemento de *orden T*. Se define la secuencia *F(1), F(2), ..., F(p-1)* por : $F(2^i u^j \bmod p) = i$ , para *i* desde *0* hasta *m-1* y *j* desde *0* hasta *T-1*.

Si $a = (a_0 a_1 a_2 ..... a_{m-1})$ y $b = (b_0 b_1 b_2 ... b_{m-1})$ son elementos de GF($2^m$), entonces

$a \bullet b = c = (c_0 c_1 c_2 ... c_{m-1})$, donde

$c_i = \sum_{k=1}^{p-2} a_{F(K+1)} \bullet b_{F(P-K)}$

- *Inversión* :

si $a \neq 0$ y $a \in$ GF($2^m$), la inversa de *a* denotado como $a^{-1}$ es el único elemento $c \in$ GF($2^m$) para lo cual $a \bullet c = 1$.

El algoritmo usado para inversión esta basado en la identidad:

$$a^{-1} = a^{2^n - 2} = \left(a^{2^{n-1}-1}\right)^2$$

En [9], Itoh y Tsujii propusieron un método que minimiza el numero de multiplicaciones para calcular la inversiones, el cual esta basado en las siguientes identidades:

$$a^{2^{n-1}-1} = \begin{cases} (a^{2^{\frac{n-1}{2}}-1})^{2^{\frac{n-1}{2}}} \bullet a^{2^{\frac{n-1}{2}}-1}, impar \\ a\left(a^{2^{n-2}-1}\right)^2, par \end{cases}$$

## 3. ALGORITMO DE MULTIPLICACIÓN PARA BASES NORMALES GAUSSIANAS EN GF($2^{163}$)

La multiplicación es la operación fundamental para la implementación de sistemas criptográficos basados en curvas elípticas. Entonces, con el propósito de alcanzar una buena seguridad para la información, NIST ha recomendado como un estándar el cuerpo finito GF($2^{163}$) para ser utilizados en los sistemas criptográficos basados en curvas elípticas.

Debido a que la multiplicación usando bases normales optimas no incluye el cuerpo finito GF($2^{163}$), el algoritmo de multiplicación usando bases normales gaussianas presenta una mayor complejidad computacional.

Un algoritmo a nivel de bits para la multiplicación en el cuerpo finito binario GF($2^{163}$) usando bases normales gaussinas es presentado en [6][7], el cual es descrito en las Tablas 1 y 2. En este caso se definen los siguientes parámetros:

| | | |
|---|---|---|
| m | = 163, | número de bits |
| T | = 4, | número recomendado por NIST para el cuerpo finito GF($2^{163}$) |
| p | = 653, | numero primo p = Tm + 1 |
| U | = 149, | numero que satisface la relación $U^4 \bmod p = 1$ |

Algoritmo 1. Multiplicación para BNG en GF($2^m$).

***Entrada*** : operandos A,B ***Salida*** : producto C

1. U ← A = ($a_0$, $a_1$, ...$a_{m-1}$)
2. V ← B = ($b_0$, $b_1$, ...$b_{m-1}$)
3. ***para*** k = 0 ***hasta*** m-1 realizar:
4. $c_{(k)}$ = F(U,V)
5. U = rotación a la izquierda de U
6. V = rotación a la izquierda de V
7. ***fin***
8. C ← C = ($c_0$, $c_1$,... $c_{m-1}$)

Donde:

F(U,V) = $\sum_{k=1}^{p-2} U_{j(k+1)} V_{j(k)}$

**Tabla 1**. Algoritmo de multiplicación en GF($2^{163}$) usando bases normales gaussianas

## 4. ARQUITECTURA DEL MULTIPLICADOR PARA BASES NORMALES GAUSSIANAS EN GF($2^{163}$)

En esta sección se presenta el diseño de una arquitectura para un multiplicador en el cuerpo finito GF($2^{163}$) usando bases normales gaussianas. La arquitectura presenta cuatro bloques funcionales, los cuales permiten implementar las diferentes funciones para llevar acabo la multiplicación. El

diagrama de bloques del multiplicador se muestra en la Figura 1.

Algoritmo 2. Generación de los subíndices j(k) de la función F(U,V).

***Entrada***: parámetros T, m, U, p ***Salida***: j(1), j(2),... j(p-1)

1. w ← 1
2. ***para*** j = 0 ***hasta*** T-1 realizar
3. n ← w
4. ***para*** i = 0 ***hasta*** m-1 realizar
5. j(n) ← i
6. n ← 2n mod p
7. ***fin***
8. w ← UW mod p
9. ***fin***

**Tabla 2**. Algoritmo de generación de J(k)

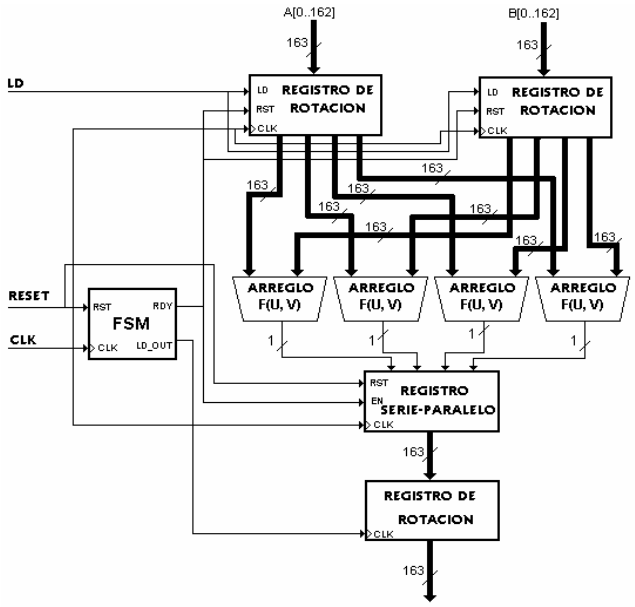



**Figura 1**. Arquitectura del multiplicador para bases normales gaussianas en GF($2^{163}$)

### 4.1 Arreglo para la función F(U,V)

El arreglo para la función F(U,V) es el bloque funcional mas importante en la arquitectura del multiplicador, y es un arreglo de compuertas *AND* y *XOR,* el cual permite implementar la función F(U,V) del algoritmo presentado en la sección anterior. La descripción funcional del arreglo para la función F(U, V) es:

$C_0 <= ( U_0 V_1 ) \oplus ( U_1 V_0 ) \oplus ( U_1 V_{117} ) \oplus ( U_1 V_{13} ) \oplus U_{102} V_{110} ) \oplus .....$
$( U_{97} V_{149} ) \oplus ( U_{97} V_{42} ) \oplus .....$
$( U_{99} V_{115} ) \oplus ( U_{99} V_4 ) \oplus ( U_{99} V_{58} ) \oplus ( U_{99} V_{90} )$

La gran particularidad del arreglo es que presenta una estructura irregular, y por lo tanto el hardware es dedicado para implementar el cuerpo finito GF($2^{163}$). En este caso, no es posible desarrollar una arquitectura sistólica o estructurada para hacer una descripción VHDL de tipo genérico, es decir, que con un pequeño cambio en la asignación del tamaño de la palabra de datos de entrada (*m)*, se puede generar un arreglo para la función F(U, V) para diferentes tamaños de palabra. El arreglo tiene dos entradas de datos de 163 bits cada una y una salida de 1 bit. Todos los bits de *U y V* entran en forma paralela al arreglo para la función F(U, V) y se obtiene solamente un bit $c_i$ de salida, el bit $c_i$ de salida del arreglo es almacenado y desplazado en el registro serie-paralelo. Posteriormente los datos U y V son rotados para obtener un nuevo bit $c_i$, esta secuencia se repite 163 veces hasta obtener el producto A.B = C .

De acuerdo a la descripción funcional, el arreglo para la función F(U,V) esta compuesto por 652 compuertas AND y 651 compuertas XOR. Con el propósito de reducir el numero de compuertas, algunas modificaciones fueron llevadas a cabo, lo cual permite obtener 644 compuertas AND y 643 compuertas XOR.

El hardware del arreglo para generar la función F(U,V) se muestra en la Figura 2.

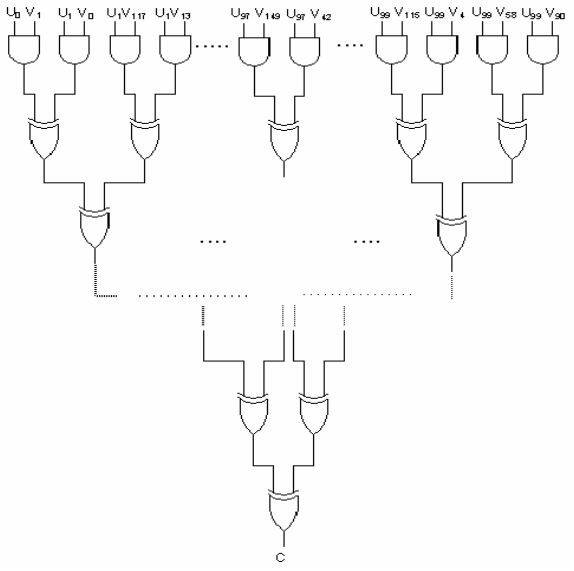



**Figura 2**. Hardware del arreglo para generar la función F(U,V)

### 4.2. Registros de rotación

Los registros de rotación son los registros encargados de desplazar y rotar los datos de entrada U y V para el arreglo de la función F(U,V).

### 4.3 Circuito fin de multiplicación

El circuito fin de multiplicación es un circuito contador, el cual cuenta el numero de rotaciones que llevan acabo los registros de rotación, y cuando la cuenta llega a 163 rotaciones, el circuito genera una señal para indicar el fin de la multiplicación.

### 4.4 Registro serie-paralelo

El registro serie-paralelo captura en cada ciclo de reloj el bit de salida del arreglo de la función F(U,V) y este bit es almacenado en un registro de 163 bits, es decir, cada vez que se captura el bit de salida, los bits anteriores son desplazados, en otras palabras este registro convierte la información de serie a paralelo, y de esta forma el resultado es almacenado en el registro de 163 bits.

### 4.5 Unidad de control

La unidad de control es implementada usando una maquina de estados finito (FSM), la cual establece la secuencia del procesamiento de los datos.

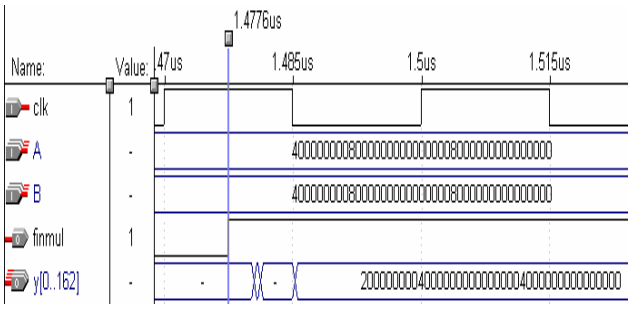
## 5. RESULTADOS DE SIMULACIÓN

Una vez diseñada la arquitectura del multiplicador, cada bloque funcional del multiplicador es descrito de manera estructural usando VHDL. Posteriormente, esta arquitectura es sintetizada sobre el circuito CPLD EPF10K200SFC672-1X y se llevan acabo las respectivas simulaciones con el propósito de verificar el funcionamiento del multiplicador. Los resultados de las simulaciones para las principales operaciones de multiplicación son mostradas en la Figura 3.

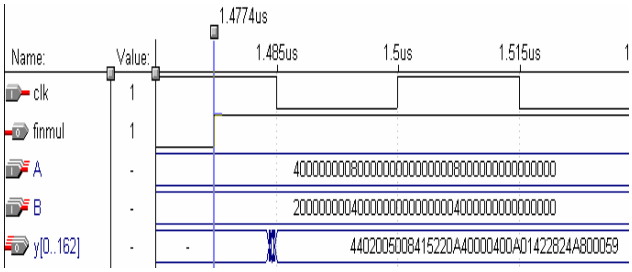
Desde los resultados de la simulación se observa que el periodo de reloj es de 30ns, sin embargo, el mínimo periodo de reloj con el cual trabaja el circuito es de 23.5ns, lo cual implica que la máxima frecuencia de la señal de reloj es de 42.55Mhz.

En la Tabla 3 se muestran los resultados de los diferentes diseños propuestos para la arquitectura del multiplicador, presentando el número de celdas lógicas utilizadas en el CPLD EPF10K200SFC672-1X, y el tiempo que tardan en realizar una multiplicación; cada diseño difiere en el numero de arreglos F(U, V) así: el diseño 1 tiene un arreglo f(U, V), el diseño 2 dispone de dos arreglos y por ultimo el diseño 3 tiene 4 arreglos F(U, V) tal como se observa en la Figura 1. Estos resultados se obtuvieron para un periodo de reloj de 30ns.

$a.a = a^2 = b.$




$a.b = a^3 = c.$




$a.c = b^2 = a^4.$

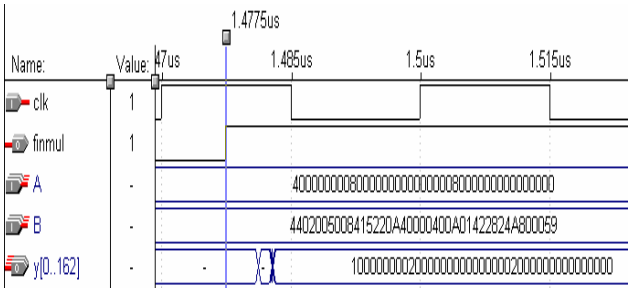



**Figura 3**. Resultados de simulación para las operaciones $a^2$, $a^3$ y $a^4$

| | Celdas lógicas | % de celdas lógicas | Tiempo de multiplicación en µs |
|---|---|---|---|
| Diseño 1 | 2214 | 22 | 4.9584 |
| Diseño 2 | 2647 | 26 | 2.5283 |
| Diseño 3 | 3839 | 34 | 1.2974 |

**Tabla 3**. Tiempo de multiplicación y número de celdas lógicas para diferentes arquitecturas

## 6. CONCLUSIONES

El diseño a nivel de hardware de un nuevo multiplicador en el cuerpo finito binario GF($2^{163}$) usando bases normales gaussianas es presentado. En este trabajo, tres diferentes diseños para la arquitectura del multiplicador han sido propuestos. En este caso, cada diseño se ha implementado (simplificación del arreglo para la función F(U,V) y duplicación del hardware para F(U,V)) con el propósito de alcanzar un mayor desempeño a nivel de área y velocidad.

Debido a la irregularidad de la estructura del arreglo de la función F(U,V) no es posible diseñar una arquitectura estructurada del multiplicador para cualquier valor de *m* del cuerpo finito GF($2^m$) usando bases normales gaussianas. Sin embargo, debido a que la descripción es realizada en VHDL y la implementación del multiplicador es llevada acabo usando circuitos CPLDs-FPGAs, es importante concluir que a partir de este primer diseño es posible diseñar rápidamente otros multiplicadores para cualquier valor de m del cuerpo finito GF($2^m$) usando bases normales gaussianas.

El trabajo futuro esta orientado a diseñar un multiplicador en cuerpo finito GF($2^{233}$) usando bases normales gaussianas, y al diseño de un criptoprocesador para un criptosistema basado en curvas elípticas.

## 7. AGRADECIMIENTOS

Este trabajo ha sido patrocinado por Altera Corporation a través del Programa Universitario. Los autores dan especial agradecimientos a Mrs. Ralene Marcoccia de Altera Corporation.

## 8. BIBLIOGRAFÍA

[1] L. Song and K.K. Parhí, "Low energy digit-serial/parallel finite field multipliers", Journal of VLSI Signal Processing Systems, 1997.

[2] L. Gao, S. Shrivastava, and G. Sobelman, "Elliptic curve scalar multiplier design using FPGAs", Workshop on cryptographic hardware and embedded systems (CHES'99), 1999.

[3] G. Orlando and C. Paar, "A super-serial Galois field multiplier for FPGAs and its application to public key algorithms", IEEE symposium on field-programmable custom computing machines (FCCM'99), 1999.

[4] C. Kim, S. Oh and L. Jongin, "A new hardware architecture for operations in GF($2^m$)", IEEE Transactions on Computers, 2002.

[5] J. Goodman and A. Chandrakasan, "An energy-efficient reconfigurable public-key cryptography processor", IEEE Journal of solid-state circuits, 2001.

[6] Digital Signature Standard, FIPS PUB 186-2, January 2000.

[7] J. Solinas, "Efficient arithmetic on Koblitz curves", Design, codes and Cryptography, 2000, 195-249

[8] D. Johnson and A. Menezes, "The elliptic curve digital signature algorithm (ECDSA)", Technical report CORR 99-34, University of Waterloo, 2000

[9] T. Itoh y S. Tsujii, "A fast algorithm for computing multiplicative inverses in GF($2^m$) using normal bases", Information and Computation, 1988.