IMPLEMENTACIÓN DE LA MULTIPLICACIÓN MODULAR DE MONTGOMERY EN HARDWARE REPROGRAMABLE

Rubén Darío Palacios Rivas, Álvaro Bernal Noreña, Rubén Darío Nieto Londoño

Escuela de Ingeniería Eléctrica y Electrónica
Grupo de Arquitecturas Digitales y Microelectrónica
Universidad del Valle
Cali - Colombia

rdpalr@hotmail.com
rnieto@eiee.univalle.edu.co
alvaro@mafalda.univalle.edu.co

RESUMEN

La multiplicación modular es requerida para la ejecución de la exponenciación modular, la cual se muestra como la operación de mayor aparición en los protocolos de aplicación criptográfica. En este artículo se presenta un diseño hardware que permite la rápida evaluación de multiplicaciones modulares con base en el algoritmo de Montgomery para números de n bits. Se obtuvieron resultados para operandos de 64 y 128 bits, en un dispositivo lógico programable, específicamente un CPLD de Altera Corp.

ABSTRACT

Modular exponentiation is widely used in several cryptographic protocols. Modular exponentiation is executed as a repetition of several modular multiplications. Architecture's performance for executing this kind of operations must be improved in order to obtain higher operations time. In this article hardware for the fast evaluation of modular multiplication based on Montgomery's algorithm is presented. The results were obtained for operands of 64 and 128 bits, using a logical device reprogrammable more specifically an CPLD.

IMPLEMENTACIÓN DE LA MULTIPLICACIÓN MODULAR DE MONTGOMERY EN HARDWARE REPROGRAMABLE

Rubén D. Palacios R., Rubén D. Nieto Londoño, Álvaro Bernal N.

Escuela de Ingeniería Eléctrica y Electrónica
Grupo de Arquitecturas Digitales y Microelectrónica
Universidad del Valle
Cali – Colombia

RESUMEN

En este artículo se presenta el diseño de hardware que permite la rápida evaluación de multiplicaciones modulares con base en el algoritmo de Montgomery para números de n bits. Se obtuvieron resultados para operandos de 64 y 128 bits, en un dispositivo lógico reprogramable, específicamente un CPLD de Altera Corp.

Palabras clave

Aritmética modular, Técnicas criptográficas, CPLD, algoritmo de Montgomery.

Keywords

Modular multiplication, CPLD, Montgomery's algorithm.

ABSTRACT

In this article hardware for the fast evaluation of modular multiplication based on Montgomery's algorithm is presented. The results were obtained for operands of 64 and 128 bits, using a logical device reprogramable more specifically an CPLD.

1. INTRODUCCIÓN

Dada la importancia que tiene el intercambio de información en el mundo moderno, las técnicas criptográficas han adquirido mucha trascendencia en los sistemas de comunicación.

El objetivo de cada una de estas técnicas es proteger el contenido de la información que se intercambia entre dos o más entidades, y también garantizar la legitimidad de las entidades que intervienen en el proceso. Para lograr tal objetivo, las técnicas criptográficas se valen de funciones matemáticas especiales tales como las operaciones modulares, las curvas elípticas y las curvas logarítmicas, entre otras.

La aritmética modular se usa por ejemplo, en protocolos como RSA [1], ElGamal [2], RPK [3], Guillou-Quisquater [4], y Fiat-Shamir [5]. Por ello existe un interés creciente en desarrollar formas rápidas y eficientes de calcular operaciones modulares binarias con operandos de gran tamaño ya que en estos protocolos el tamaño de los operandos binarios determina el nivel de seguridad del sistema.

Ha sido una práctica frecuente la ejecución de las operaciones modulares en software. Si bien una implementación en hardware puede mejorar la velocidad del sistema, su uso no es común debido a la poca flexibilidad que ofrecen este tipo de implementaciones y las herramientas tradicionales de diseño.

Sin embargo, con las ventajas ofrecidas por los lenguajes de descripción de hardware como el VHDL y de dispositivos lógicos reprogramables, es posible implementar aplicaciones en hardware de una manera muy cómoda y flexible [6]. Dadas las características de dispositivos como las FPGAs, es posible cambiar completamente una aplicación sin tener que cambiar el hardware involucrado. El VHDL permite además la descripción de hardware con herramientas de alto nivel.

En este artículo se presenta el diseño de un multiplicador modular genérico, que puede ser utilizado como parte de la implementación del protocolo criptográfico RSA para el cálculo de la exponenciación modular.

2. ALGORITMO DE MONTGOMERY

El producto modular en módulo M de dos números X e Y se realiza calculando el producto de los dos números

y luego calculando el residuo de dividir este producto por M de acuerdo con la ecuación (1).

$$C = XY \bmod M \qquad (1)$$

Sin embargo, esta forma de calcularla resulta poco práctica si el tamaño de los operandos es considerable, tal como lo requiere el protocolo RSA para obtener grados de seguridad confiables. Si se van a multiplicar dos números de n bits, se necesitarán 2n bits para el almacenamiento del resultado intermedio, además, el cálculo posterior del residuo resulta demasiado lento.

Se han desarrollado técnicas que tratan de evitar los problemas mencionados anteriormente. Uno de los algoritmos más utilizados es el de Montgomery [7], el cual se utiliza en el diseño descrito en este artículo.

El algoritmo para la multiplicación modular utilizado en este artículo fué propuesto por P.L. Montgomery en 1985 [8]. Este se basa en el cambio de residuos de *(XmodM*) en *(XRmodM*), para algunos valores de *R* escogidos de tal manera que las operaciónes *mod R* y *div R* son más fáciles de calcular que *mod M* y *div M* respectivamente.

Usualmente *R* se escoge como una potencia de la base en la que se representan estos números. Así, si la base es binaria entonces $R = 2^n$. Ahora sí *X* e *Y* son los operandos para los que las operaciones *XRmod M* y *YRmod M* son valores conocidos, entonces la multiplicación de Montgomery definida como $MM(t)=(tR^{-1} \bmod M)$ se puede utilizar para calcular *XYRmodM* solamente empleando operaciones *modR* y *div R*.

La operación *(XmodM)(YmodM))* produce $XYR^{-1} modM$, lo que permite recuperar *XYmodM* a partir de una nueva multiplicación de Montgomery entre $XYR^{-1} modM$ y la constante precalculada $R^2 modM$.

Este último hecho hace que la eficiencia de este algoritmo solo es apreciable para números de tamaño considerable y para una cantidad apreciable de productos modulares, como es el caso de la exponenciación modular.

El algoritmo básico se muestra a continuación:

Algoritmo 1: Multiplicación Modular de Montgomery (base 2) para calcular

$$X \bullet Y \bmod M$$

Donde:

$$X = \sum_{i=0}^{n-1} x_i 2^i, b_i \in \{0,1\}$$

$$Y = \sum_{i=0}^{n-1} y_i 2^i, b_i \in \{0,1\}$$

$$M = \sum_{i=0}^{n-1} m_i 2^i, b_i \in \{0,1\}$$

$$S_0 = 0$$

Para i=0 hasta (n-1) haga

Si $= (S_i + X(i) * Y)$ es par.

Entonces $S_{i+1} = (S_i + X(i) * Y) \ \text{div} \ 2$

sino $S_{i+1} = (S_i + X(i) * Y + M) \ \text{div} \ 2$

Siendo *S* el registro de iteración donde se van almacenando los resultados parciales del producto (y el resultado final de la operación), M es el módulo, X e Y son las palabras a multiplicar y n es el número de bits de los operandos. Las variables con subíndices indican el dígito correspondiente.

El módulo debe cumplir las siguientes condiciones:

$$2^{n-1} \le M < 2^n$$

M>1 y debe ser un número impar

Después de una primera ejecución del algoritmo, el valor almacenado en *S* resulta ser:

$$S = XYR^{-1} \bmod M \qquad (2)$$

Para que una multiplicación Montgomery equivalga a una multiplicación modular estándar se debe tomar el resultado obtenido en (2) y realimentarse como una entrada para una nueva ejecución del algoritmo, la otra entrada debe ser:

$$R^2 \bmod M \qquad (3)$$

Dado que los operandos del multiplicador son números binarios de n bits entonces $R^2 mod\ M = (2^n)^2 modM$ igual a $2^{2n} modM$. Una vez ejecutados los dos productos de Montgomery en forma sucesiva, el valor almacenado en *S* es el producto modular natural XYmodM.

El algoritmo de Montgomery es particularmente eficiente cuando se deben calcular una cantidad

considerable de productos, como en el caso de una exponenciación modular. Esto se debe a que el cálculo de la constante R^2modM solo debe hacerse una vez para un módulo dado. Otra de las ventajas del algoritmo es que la variable *s* donde se almacenan los resultados intermedios, debe tener un tamaño de n+1 bits, en comparación con los 2n bits mencionados anteriormente.

3. DESCRIPCIÓN DE LA ARQUITECURA.

A fin de implementar la multiplicación modular la arquitectura requiere de un sumador de tres operandos, el cual usa registros acumuladores con el fin de acelerar el proceso de realimentación para las sumas parciales, además de un divisor por 2. En el diagrama de bloques mostrado en la figura 1 se presenta una primera aproximación del hardware combinatorio usado en el multiplicador de Montgomery.

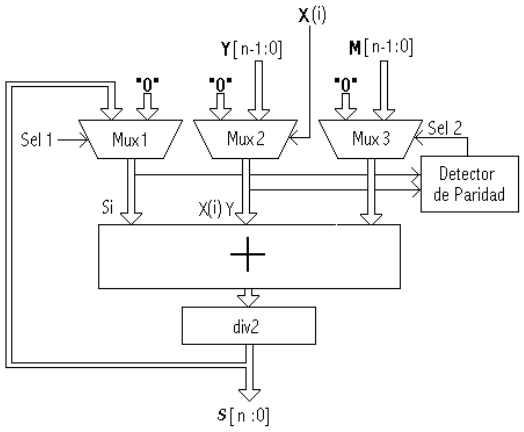

Figura 1. Diagrama de bloques del multiplicador de Montgomery para operandos de n bits.

Los bloques MUX1, MUX2, y MUX3 representan los multiplexores que se encargan de seleccionar los operandos que serán llevados al sumador. Las salidas de los bloques MUX sirven como entradas a un sumador de tres operandos (uno de los operandos es de n+1 bits y los otros dos son de n bits). La salida del sumador, de n+2 bits, sirve como entrada a un bloque div2 que se encarga de realizar la división por dos de cada suma parcial.

Un análisis detallado de la figura 1 permite observar que el bloque sumador es el que provoca más retardo en el hardware combinatorio, por tal razón el bloque sumador será el que limite la frecuencia máxima de operación del multiplicador. Con el lenguaje descripción de hardware VHDL se implementaron distintos sumadores, para luego comparar su desempeño y escoger el más adecuado para realizar el multiplicador.

Entre las diversas implementaciones del sumador de tres operandos que se describieron y simularon en VHDL, se pueden mencionar las siguientes:

- Sumador funcional[1].
- Sumador con dos niveles de acarreo propagado.
- Sumador con dos niveles de acarreo anticipado cada n bits.
- Sumador con acarreo guardado (CSA) en la primera etapa y acarreo anticipado cada n bits en la segunda.

De los sumadores presentados se escogió la última alternativa por ser la más rápida de todas. En la figura 2 se presenta la arquitectura detallada propuesta para realizar la multiplicación modular de Montgomery para operandos de n bits.

Como se puede observar, el sistema consta de 8 bloques fundamentales siendo el primero, el conformado por los registros, los cuales se encargan de retener temporalmente los datos de entrada a los bloques Sum1, Mux3, Mux4, Res1, Res2 y CPS1B. El segundo, denominado ***bloque combinatorio*** está conformado por los multiplexores Mux3 y Mux4 , el detector de paridad, el sumador de tres operandos y el divisor por dos, que se encargan de realizar las sumas parciales (S_{i+1}), determinar el estado de paridad de las sumas (S_i +X(i)*Y) y con base en esta información decidir si se agrega o nó el módulo M a la próxima suma parcial (S_{i+1}).

El tercero está compuesto por el MUX1 que al inicio del primer ciclo de ejecución del algoritmo entrega al sumador, incluido dentro de la parte combinatoria, la primera suma parcial con valor igual a cero (S_o=0) de acuerdo con el algoritmo de Montgomery y posteriormente, permitir la retroalimentación entre la salida del bloque combinatorio y una de sus entradas al cambiar el estado el bit de selección del multiplexor (Sel 1). El cuarto, denominado CPS1B es básicamente un conversor paralelo a serial cuya entrada es la palabra X de 32 bits (durante la primera ejecución del producto modular de Montgomery) y R^2modM (durante la segunda ejecución) y cuya salida es un flujo de 1 bit. A este conversor se le ha integrado un contador de 32 estados el cual es necesario para controlar el número de iteraciones.

El quinto, es el multiplexor de 4 entradas y dos salidas de 32 bits denominado MUX2. que se encarga de entregar al resto del hardware los operandos correctos

[1] En la descripción de este sumador se aprovecha las herramientas de alto nivel que ofrece el lenguaje VHDL para la descripción de hardware. Se está aprovechando una de las características de la POO llamada sobrecarga. Por medio de una de las librerías de la IEEE es posible sobrecargar al operador ´+´ del VHDL para que sume vectores de bits como si fueran números enteros. La sobrecarga del operador le permite al diseñador abstraerse de la implementación física del sumador y concentrarse en la funcionalidad del mismo

para la realización de cada producto modular no estándar mediante el control del nivel de la señal de selección Sel2. Cuando el nivel de esta señal es bajo el multiplexor MUX2 entrega en su salida los operandos X e Y. Cuando es alto las salidas son XYR^{-1}modM y R^2mod M respectivamente. El sexto, lo constituye el bloque del Res1, el multiplexor MUX5 y la compuerta AND de dos entradas.

Las estructuras mencionadas anteriormente realizan esta función de ajuste del resultado. Para ello, el bloque Res1 se encarga de hacer la resta entre las señales Sum y M. La señal Sum (que es de n+1 bits) es la salida del multiplicador de Montgomery sin ajustar.

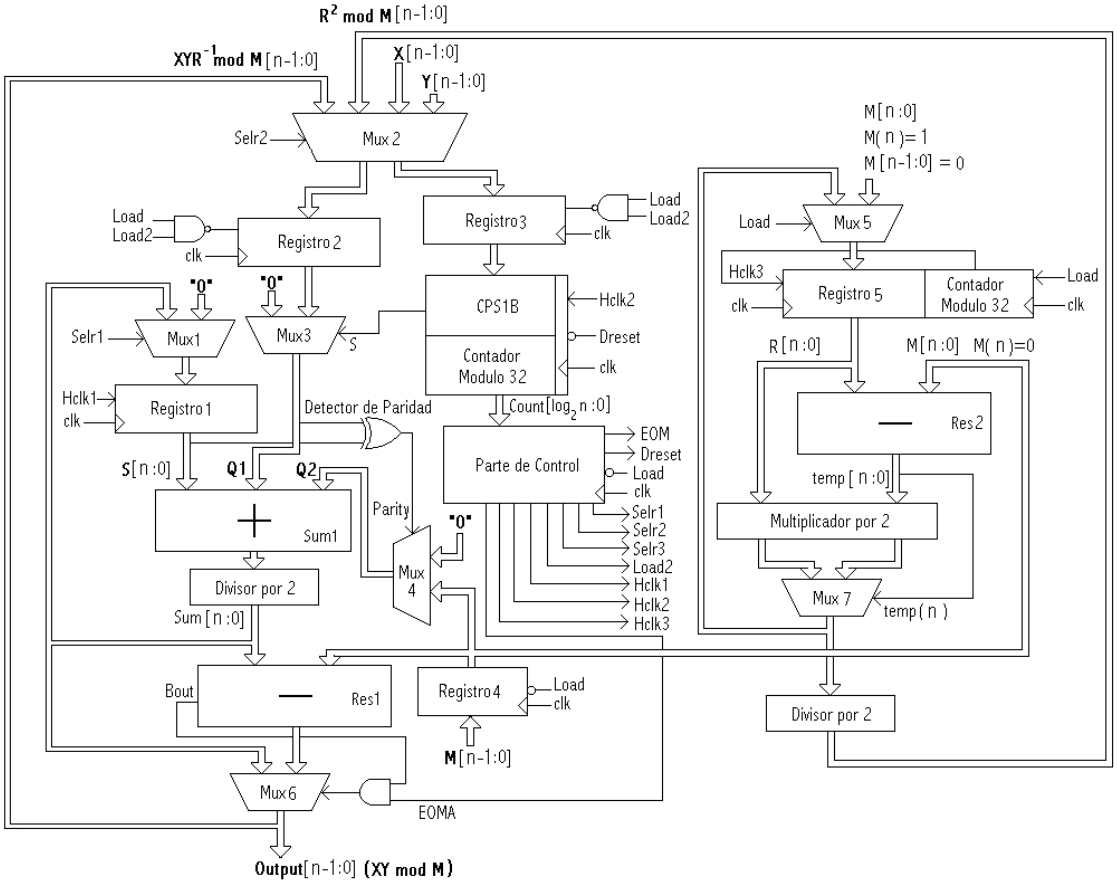

Figura 2. Diagrama esquemático del multiplicador de Montgomery .

Este bloque es crucial debido a que el algoritmo de de Montgomery en algunas ocasiones entrega como resultado el valor esperado de la operación sumado al módulo de la misma. Esto se debe a que en aritmética módulo M un número k es equivalente a los números k + M, o también k + 2M, k + 3M,... etc. Lo que puede implicar que el resultado obtenido podría requerir n+1 bits. Por estas dos razones el resultado debe ser ajustado a n bits con el fin de permitir la adecuada realimentación para la segunda ejecución del algoritmo. Esto se puede hacer fácilmente verificando si el resultado del producto de Montgomery es menor al módulo M. Si es menor significa que el resultado no necesita ningún ajuste. En caso contrario deberá restarse al resultado el valor de M.

La señal M por su parte es el módulo de la operación compuesta por n bits. La señal Bout representa el préstamo (borrow) de salida en la operación de resta. Si Bout es cero significa que el operando Sum es mayor o igual al operando M. Si Bout es uno significa que el operando M es mayor a Sum.

La señal EOMA que también está involucrada en este proceso y que es una de las salidas que tiene la unidad de control, se pone en nivel alto cuando finaliza cada producto modular, indicando el momento en el que se debe realizar el ajuste de los resultados (recuérdese que para obtener el producto modular completo es necesario realizar dos multiplicaciones de Montgomery). Considerando esto, la operación AND entre las señales Bout y

EOMA se usa como entrada de selección en el multiplexor MUX6 que decide si el resultado ajustado del producto de Montgomery corresponderá a la resta entre Sum y M ó a Sum simplemente. La señal Output corresponde al producto de Montgomery ajustado. El séptimo, conformado por Mux 5, Mux 7, registro 5 y Res2 evalúan la constante R^2modM necesaria para obtener el producto modular completo.

Finalmente se encuentra la unidad de control. Esta genera las señales de sincronización para todo el sistema. Una señal de salida (EOM) indica el momento en que el producto modular de Montgomery se obtiene completamente.

En la figura 3 se aprecia un diagrama más detallado del ***bloque combinatorio*** definido anteriormente. Como se puede observar, consta de un sumador de tres operandos tipo CSA.

Un multiplexor (MUX2) en donde una de sus entradas (que es interna) adopta el valor de cero de n bits y la otra (que es externa) adopta el valor de la palabra Y de n bits, tiene una señal de selección que se alimenta por la salida serial del bloque CPS1B lo que en definitiva permite deducir que en la salida de este multiplexor se obtiene el producto X(i)*Y, es decir si X(i)=0 la salida de MUX2 es cero, si es lo contrario entonces el valor de la salida de MUX2 es X(i)*Y.

La señal de salida de la compuerta XOR (detector de paridad) indicará con un nivel alto si la suma S_i+X(i)*Y es impar y con un nivel bajo si es par al sensar con sus dos únicas entradas los primeros bits del registro de iteración S y de X(i)*Y. Con el valor obtenido en la salida de este detector, la señal de selección de MUX3 permitirá o no el paso del modulo M al sumador. Después de realizar la suma global, el resultado se divide por dos.

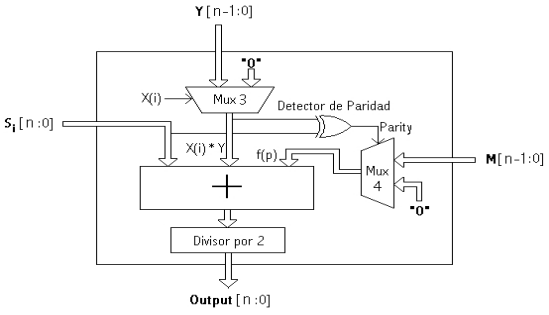

Figura 3. Diagrama del bloque combinatorio.

Se evaluó otra alternativa a la arquitectura que se ha venido discutiendo. Como se explicó anteriormente, cada iteración dentro del algoritmo de Montgomery implica la suma de tres operandos condicionada por ciertas restricciones.

La nueva arquitectura, realiza cuatro de estas sumas por cada iteración. En otras palabras, se requerirán 4 veces menos iteraciones para el cálculo del producto modular al utilizar este diseño. Para poder lograr esto fue necesario conectar en serie cuatro bloques combinatorios como el descrito en la figura 3, además de una variación en el bloque CPS1B para realizar una conversión paralela a serial de n a 4 bits. Con estas variaciones el multiplicador se puede utilizar para operandos que sean múltiplos de 4. Obviamente la complejidad agregada para poder lograr esto, hace que el retardo del hardware combinatorio aumente en forma considerable. El objetivo de esta implementación es averiguar si el compromiso entre el retardo agregado en el hardware y la reducción del número de iteraciones permite aumentar la velocidad de los cálculos.

4. RESULTADOS

La tabla 1 ilustra los resultados que se obtuvieron para los prototipos de 64 y 128 bits en un solo dispositivo lógico programable. Para la simulación se usó el paquete MAX+plus II versión 10.0 de Altera [9]. Todos los diseños se simularon en el CPLD de Altera EP1K100FC484-1 que posee 4992 celdas lógicas. La información obtenida de los diseños se hizo a partir de cinco parámetros.

El primero corresponde a la máxima frecuencia de operación del sistema. El segundo, el tiempo de cálculo promedio. El tercero, el tiempo de máximo retardo. El cuarto y el quinto hacen referencia a la cantidad de celdas lógicas y flip-flops utilizados en la implementación.

En las figuras 4, 5 y 6 se observan los gráficos de la respuesta típica de los multiplicadores. En el caso de 128 bits, se tomaron como operandos de entrada los siguientes:

X=C12345AB1025BF05C12345AB1025BF05.

Y=B4512AAABBBB00CC12345678B4512AA.

M=FFFF0000FFFFFFFFFFFFFFFFFFFFFFFF.

Tabla 1. Resultados de las simulaciones de los multiplicadores de 64 y 128 bits en la familia ACEX1K de Altera.

DISEÑO: ASM64ECPS4B								
		RESPUESTA EN FRECUENCIA			Línea de mayor retardo. Tiempo máximo de retardo. [nS]		Número de Celdas lógicas requeridas	Nro. de Flip-Flops
FAMILIA	DISPOSITIVO	Fmáx. [MHz]	Ciclo periodo [nS]	Tiempo de Calculo promedio [µS]				
ACEX1K	EP1K100FC484-1	2,60	384,0	5,07	Output (1)	:151,6	2488/4992	340
DISEÑO: ASM128ECPS4B								

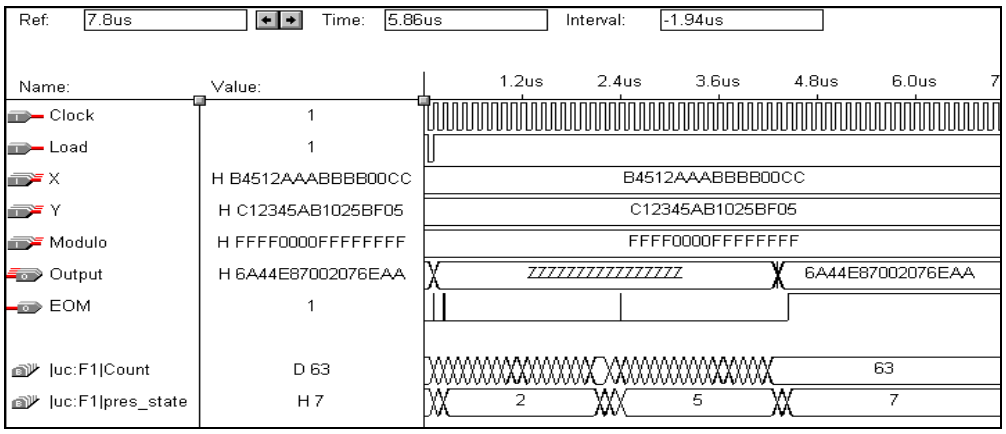

Figura 4. Resultados del multiplicador de 64 bits

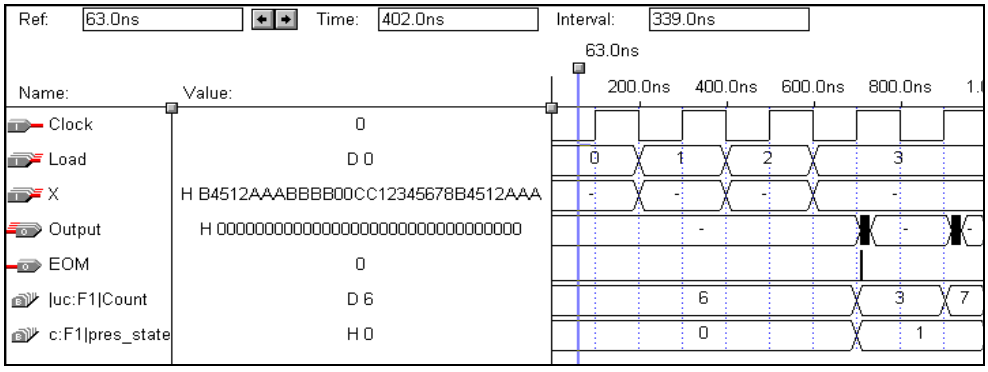

Figura 5. Proceso de carga de los operandos para el multiplicador de 128 bits.

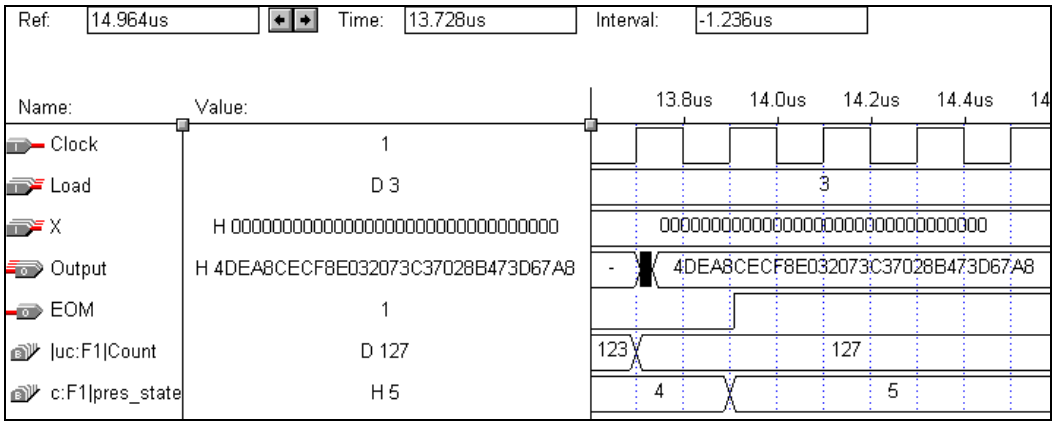

Figura 6. Resultado de final de la multiplicación para 128 bits.

El producto modular vendría dado por la señal *Output* y sería igual a:

4DEA8CECF8E032073C37028B473D67A8

Todos los operandos en formato hexadecimal.

5. CONCLUSIONES

A lo largo de este articulo se evidenció la importancia que tiene en el diseño de hardware para técnicas criptográficas el uso del algoritmo de Montgomery. La simplicidad de los diseños presentados (que están compuestos en su parte combinatoria por sumadores, restadores, multiplexores y divisores por 2) contrasta con las principales restricciones de este algoritmo como son la obligación de evaluar la constante $R^2 \text{mod} M$, la de realizar dos multiplicaciones modulares no estándar para entrar y salir del espacio de residuos de Montgomery así como el incremento en el tiempo de cálculo en la medida en que aumente el tamaño de los operandos.

En lo relacionado con las arquitecturas presentadas para el multiplicador modular, se puede establecer que se ha logrado un desempeño aceptable en términos de área ocupada y velocidad de ejecución teniendo en cuenta que el algoritmo de Montgomery es secuencial y que en el peor de los casos se necesitarán 2n ciclos de reloj para obtener los resultados.

Los diseños presentados en este artículo se pueden adaptar a operandos de mayor tamaño, siempre que la capacidad del CPLD lo permita.

6. REFERENCIAS

[1] R. Rivest, A. Shamir, L Adleman, A method for obtaining digital signatures and public-key cryptosystems, Communication of the ACM, 21,1978.

[2] T. ElGamal, A public key cryptosystem and a signature scheme based on discrete logarithms, IEEE Transactions of information Theory, v. IT-31, No. 4, Jul. 1985.

[3] W.M. Raike, The RPK Public key Cryptographic System, Technical summary July, 1996.

[4] J.J. Quisqater, C. Couvreur, Fast decipherment algorithm for RSA Public-key Cryptosystem, Electronics Letter, vol. 18, 1982.

[5] A. Fiat, A. Shamir, How to improve yourself: Practical solutions to identification and signature problems. Advances in Cryptology – Proc. Of Crypto'86, 1986.

[6] Bashker, J. A VHDL primer. Prentice Hall. 1995.

[7] A. Bernal, Conception Etude d′ une Architecture Numérique de Haute Perfomance pour le Calcul de la Fonction Exponentielle Modulaire.

[8] P. L. Montgomery, Modular Multiplication without trial division. Mathematics of Computation. 1985.

[9] Altera Digital Library 2001 Version 1, Altera Corporation. San José, CA.