ALGORITMO DE CIFRADO DES EN UN PROCESADOR NIOS II

Cyntia E. Enríquez Ortiz
Raúl Fernández Zavala

Resumen. **Los algoritmos para cifrado de datos pueden ser implementados en diversas formas, la de mayor flexibilidad es probablemente una solución basada completamente en software. Esta solución presenta un alto grado de flexibilidad al costo de un bajo desempeño. Una alternativa a la implementación de algoritmos de cifrado en software es el desarrollo de circuitos integrados dedicados especialmente al cifrado de datos, sin embargo esta solución carece de la flexibilidad que ofrecen los sistemas basados en microprocesadores. Otra alternativa es la integración de instrucciones de aplicación específica dentro de un procesador de propósito general, lo cual combina la flexibilidad de una solución en software con la eficiencia y desempeño de una solución de hardware dedicado. En este trabajo se describe la arquitectura en un FPGA para la implementación de instrucciones de aplicación específicas de cifrado/descifrado DES en un procesador NIOS II.**

Palabras Clave: **DES, cifrado, FPGA, NIOS.**

I. Introducción

En la actualidad la demanda de conectividad a grandes redes de cómputo se ha extendido a sistemas tales como teléfonos celulares, asistentes personales digitales (PDA), redes de sensores y otros dispositivos móviles con capacidad de conexión inalámbrica; sin embargo la información que fluye a través de estas redes es vulnerable y puede ser monitoreada o modificada por usuarios no autorizados, por lo tanto es importante contar con mecanismos que garanticen seguridad en la transmisión de información.
Mediante la criptografía, se puede proteger la información que es enviada a través de sistemas de comunicación poco confiables, ya que permite garantizar el secreto en la comunicación entre dos entidades y por otra parte permite asegurar que la información sea auténtica, es decir, que el contenido del mensaje no ha sido modificado y que procede de la fuente correcta.

Los algoritmos de cifrado se clasifican básicamente en dos tipos: cifrado simétrico o de clave privada y cifrado asimétrico o de clave pública. El cifrado asimétrico utiliza dos claves independientes, una para el cifrado del mensaje y otra para el descifrado.

Actualmente los algoritmos de cifrado se pueden implementar en diversas formas, la más simple y de mayor flexibilidad es probablemente una solución basada completamente en software, donde uno o más procesadores programables realizan todas las funciones requeridas y además se tiene la posibilidad de cambiar el algoritmo de cifrado simplemente ejecutando un programa diferente. Esta solución presenta un alto grado de flexibilidad al costo de un bajo desempeño y baja eficiencia en consumo de potencia, por otra parte, estas implementaciones pueden ser lentas para sistemas que trasmiten grandes flujos de información o bien en equipos con bajo poder de cómputo como son los dispositivos móviles. Una alternativa a la implementación de algoritmos de cifrado en software es el desarrollo de hardware dedicado especialmente al cifrado de datos mediante circuitos integrados de

aplicación específica (*ASIC - Application Specific Integrated Circuit*), sin embargo esta solución carece de la flexibilidad que ofrecen los sistemas basados en microprocesadores. Una solución con mayor eficiencia puede ser obtenida empleando arreglos de compuertas programables en campo (*FPGA - Field Programable Gate Array*), los cuales ofrecen la capacidad de poder ser reconfigurados, brindando de esta forma flexibilidad. Una combinación de procesadores y FPGAs constituye una solución de gran flexibilidad y potencia para realizar las funciones de procesamiento habituales en los sistemas de cifrado de información. Otra alternativa en el cifrado de datos, que es el uso de procesadores de conjunto de instrucciones de aplicación específica (*ASIP - Application Specific Instruction set Processor*), los cuales combina la flexibilidad de una solución en software con la eficiencia y desempeño de una solución de hardware dedicado. Un ASIP básicamente es un procesador que ha sido diseñado especialmente para realizar cierta clase de tareas en forma eficiente y con la funcionalidad requerida. Otros enfoques adoptados recientemente para acelerar el cifrado de datos son la utilización de las Unidades de Procesamiento Gráfico y la extensión del conjunto de instrucciones de procesadores de propósito general.

II. El Algoritmo DES

El algoritmo DES [1] es un algoritmo de cifrado simétrico de bloques, en el cual el transmisor y el receptor utilizan una sola llave tanto para el cifrado como el descifrado. El tamaño del bloque de datos es de 64 bits, mientras que la longitud de la llave es de 56 bits (64 bits si se consideran 8 bits de paridad). El algoritmo es iterativo y cada iteración se denomina ronda y en forma general es la combinación de dos técnicas básicas de cifrado: confusión y difusión. El cifrado se realiza en tres fases, en la primera a un bloque de 64 bits se le aplica una permutación inicial (IP). La segunda fase consiste de 16 rondas que involucran funciones de substitución y permutación, a la vez que combina los datos con la llave. Después de la última ronda se aplica una permutación final que es el inverso de la permutación inicial. En la figura 1 se muestra el diagrama a bloques del algoritmo DES.

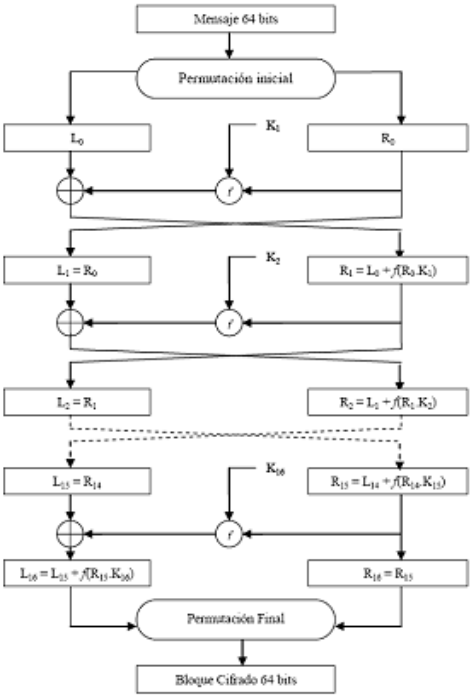

Fig. 1. Algoritmo DES

La llave de 56 bits se utiliza para generar 16 subllaves de 48 bits. Inicialmente, sobre los 56 bits de la llave se aplica una permutación (PC1), posteriormente para cada una de las 16 rondas de cifrado se producen las subllaves mediante un corrimiento a la izquierda y una permutación (PC2) que seleccionan 48 de los 56 bits de la llave desplazada. En el diagrama de la figura 2 se muestran las operaciones involucradas en una ronda de cifrado y de generación de llave.

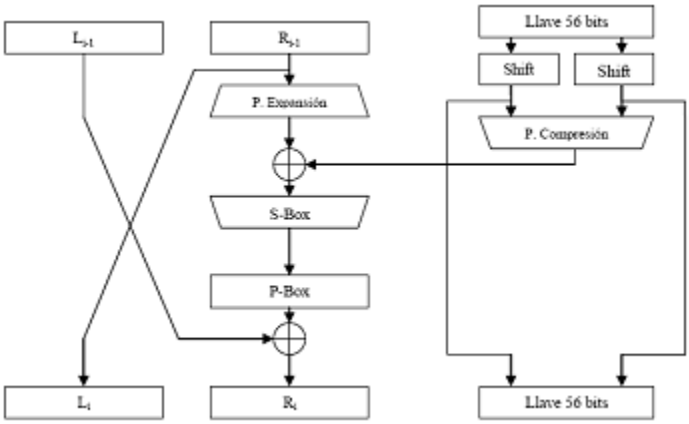

Fig. 2. Ronda del algoritmo DES

III. Implementación

El procesador Nios II es una unidad central de procesamiento configurable que se define en un lenguaje descriptor de hardware y que puede ser implementada en los Arreglos de Compuertas Programables en Campo (*FPGA Field Programmable Gate Array*) de la compañía

Altera. El núcleo del procesador NIOS II es una arquitectura *RISC (Reduced Instruction Set Computer*) tipo Harvard que se caracteriza por un conjunto de instrucciones, ruta de datos y espacio de direccionamiento de 32 bits y que cuenta con 32 registros de propósito general. Una característica distintiva de este procesador es la capacidad de permitirle al usuario definir instrucciones que se incorporan directamente a la Unidad Aritmética-Lógica (*ALU Arithmetic Logic Unit*) como se muestra en la fig. 3 y optimizan el desempeño del sistema para alguna aplicación específica. La lógica a medida (*Custom Logic*) que implementa las unidades funcionales para la extensión de instrucciones requiere dos entradas de 32 bits y genera una salida de 32 bits [2].

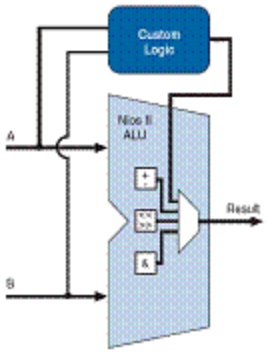

Fig. 3. Instrucciones a medida en el NIOS II [2].

Debido a que los microprocesadores de propósito general están orientados a operaciones con palabras, las operaciones a nivel de bit requieren de hasta cuatro instrucciones por bit, para una permutación arbitraria. En general una permutación arbitraria de 64 bits podría tomas hasta 128 o 256 instrucciones [3]. Otra forma de realizar una permutación es utilizando tablas, de esta forma una permutación de *n* bits requiere una tabla con 2 a la *n* localidades o bien *m* tablas con 2 a la *m/n* localidades. Por ejemplo una permutación de 64 bits se puede hacer con ocho tablas de 256 entradas de 64 bits (16 Kbytes) y requiere de 23 instrucciones. Para reducir el número de instrucciones y disminuir el uso de memoria, en este trabajo se presenta la extensión del conjunto básico de instrucciones del procesador NIOS II para realizar las operaciones de permutación del cifrado DES. También se incluyen instrucciones para realizar las operaciones de substitución y desplazamiento, con la finalidad de reducir el número de accesos a memoria.

Para el cifrado se definieron las instrucciones DESIP, DESINVIP, DESPE, DESSBOX y DESPP. Las instrucciones DESIP y DESINVIP realizan la permutación inicial y final respectivamente, cada una de estas operaciones toma 64 bits de entrada y produce una salida de 64 bits, en la figura 4 se muestra el diagrama a bloques de la estructura de la unidad funcional para estas instrucciones. Cada una de estas instrucciones debe realizarse dos veces para generar un resultado correcto de 64 bits, los primeros 32 bits se obtienen cuando n toma el valor 0 y los restantes 32 bits cuando n es igual a 1.

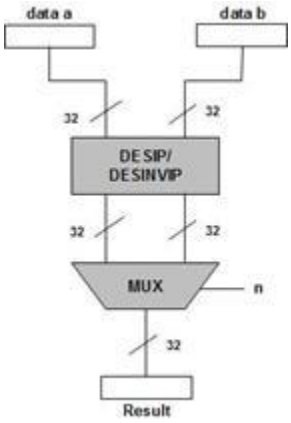

Fig. 4. Instrucción DESIP/DESINVIP

La instrucción DESPE implementa la permutación y expansión necesaria en cada ronda de cifrado. Esta operación toma como entrada un valor de 32 bits y regresa un valor con 48 bits. Al igual que la instrucción DESIP, se debe realizar dos veces para generar los 48 bits correspondientes. En la figura 5 se muestra su diagrama a bloques.

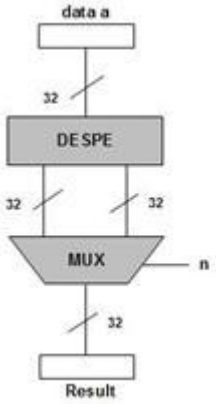

Fig. 5. Instrucción DESPE

Las cajas S (*S-Boxes*) necesarias para la instrucción DESSBOX se implementaron como 8 tablas de 64 nibbles. La principal ventaja de esta instrucción es que reduce el numero de accesos a memoria en el algoritmo DES. En la figura 6 se muestra el diagrama a bloques correspondiente a la instrucción DESSBOX.

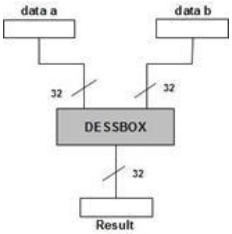

Fig. 6. Instrucción DESSBOX

En la figura 7 se muestra el diagrama a bloques de la instrucción DESPP que realiza la permutación P, como se puede observar la instrucción utiliza una entrada de 32 bits y la salida es un valor de 32 bits. En esta instrucción, al igual que la instrucción DESSBOX, no requiere del multiplexor a la salida con lo cual se lograr reducir el tiempo de ejecución a sólo un ciclo de reloj.

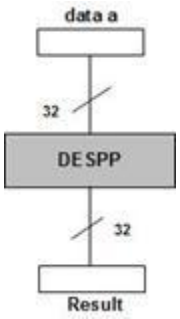

Fig. 7. Instrucción DESPP

Para la generación de las llaves se definieron las instrucciones DESPC1, DESPC2 que realizan permutaciones y DESSHIFT que produce un desplazamiento circular de uno o dos bits. En la figura 8 se muestra el diagrama a bloques de la instrucción DESPC1, la cual tiene como entrada 64 bits (56 de la llave y 8 de paridad) y genera una salida de 32 bits. La unidad funcional para la instrucción DESPPC2 tiene una estructura similar.

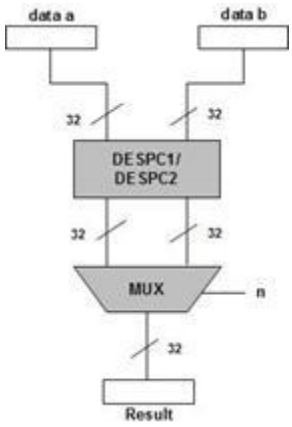

Fig. 8. Instrucción DESPC1/DESPC2

En la figura 9 se puede observar que la instrucción DESSHIFT tiene dos entradas de 32 bits, la primera de ellas (dataa) contiene 28 bits de la subllave que se va a desplazar. La segunda entrada (datab) indica el número de ronda correspondiente y cuando toma el valor 1, 2, 9 o 16 la entrada dataa se desplaza un bit a la izquierda, en tanto que para los valores restantes el desplazamiento es de dos bits.

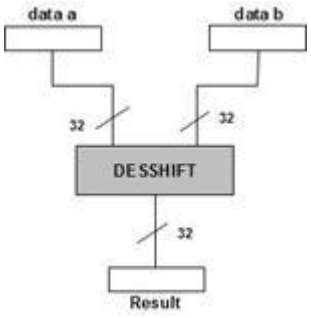

Fig. 9. Instrucción DESSHIFT

V. Validación y Resultados

La implementación del sistema se realizó utilizando un sistema de desarrollo DE2 basado en un FPGA *Cyclone II EP2C35672C6* de Altera. Como herramienta de síntesis se utilizó el ambiente de desarrollo *Quartus II Web Edition 6.0*. La descripción en hardware de las unidades funcionales para las instrucciones se realizó utilizando verilog HDL. Para sintetizar el procesador NIOS II se utilizó el programa *SOPC Builder*, el cual permite configura el núcleo del procesador y agregar diferentes periféricos y tipos de memoria para formar un sistema completo en un circuito programable (*SOPC System On a Programmable Chip*). Esta herramienta también se puede utilizar para agregar instrucciones a medida (*Custom Instructions*) en el procesador NIOS II.

Después de sintetizar y agregar a la ALU las unidades funcionales correspondientes a cada instrucción, se programó el algoritmo DES utilizando el lenguaje de programación C, para ello se utilizó el entorno de desarrollo *NIOS II IDE*. En la figura 10 se muestra en programa en C que realiza 16 rondas completas del algoritmo DES utilizando las instrucciones que realizan las permutaciones y substituciones. En este fragmento de código se pueden observar las llamadas a las funciones *ALT_CI_DESIP(), ALT_CI_DESINVIP(), ALT_CI_DESPE(), ALT_CI_DESBOX() y ALT_CI_DESPP()* las cuales en realidad son macros definidas en un archivo de cabecera y son las encargadas de invocar las instrucciones correspondientes.

```
Lo=ALT_CI_DESIP(0,DataH,DataL);
Ro=ALT_CI_DESIP(1,DataH,DataL);
for (i=0;i<16;i++) {
   Li=Lo;
   Ri=Ro;
   E1=ALT_CI_DESPE(0,Ri);
   E2=ALT_CI_DESPE(1,Ri);
   O1=E1^KH[i];
   O2=E2^KL[i];
   S=ALT_CI_DESSBOX(O1,O2);
   P=ALT_CI_DESPP(S);
   Lo=Ri; Ro=Li^P;
}
C1=ALT_CI_DESINVIP(0,Ro,Lo);
C2=ALT_CI_DESINVIP(1,Ro,Lo);
```

Fig. 10. Código en lenguaje C para una ronda de cifrado DES.

En la figura 11 se muestra el programa en C, que realiza la generación de llaves y como se puede observar llama a las funciones *ALT_CI_DESPC1(), ALT_CI_DESPC2() y ALC_CI_DESSHIFT()* dentro de un ciclo que se repite 16 veces. De igual forma se puede programar el cifrado y la generación de llaves utilizando las instrucciones a medida en lenguaje ensamblador para el procesador NIOS II, a través de la instrucción *custom,* la cual permite acceder hasta 256 instrucciones definidas por el usuario.

```
C=ALT_CI_DESPC1(0,KeyH,KeyL);
D=ALT_CI_DESPC1(1,KeyH,KeyL);
for (i=0;i<16;i++){
   C=ALT_CI_DESSHIFT(C,i);
   D=ALT_CI_DESSHIFT(D,i);
   KH[i]=ALT_CI_DESPC2(0,C,D);
   KL[i]=ALT_CI_DESPC2(1,C,D);
}
```

Fig. 11. Código en lenguaje C para la generación de llaves.

En la figura 12 se muestra la consola que utiliza el ambiente *NIOS II IDE* para comunicarse con la tarjeta de desarrollo DE2 a través de una interfaz USB. En esta figura se puede observar el resultado del cifrado del bloque de datos 0x68652074696D6520 cuando se utiliza como clave de cifrado 0x012345678ABCDEF.

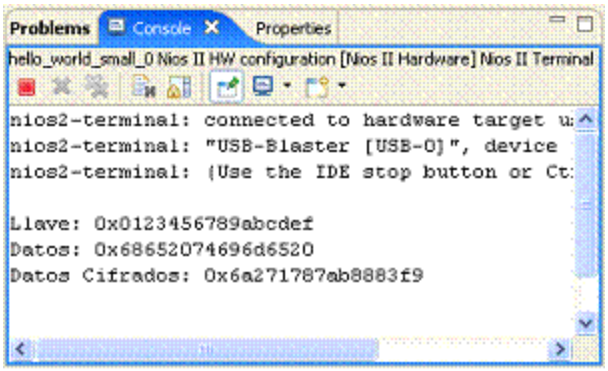

Fig. 12. Resultado del cifrado DES.

V. Conclusiones

En este trabajo se llevó a cabo el diseño hardware/software para la implementación del algoritmo de cifrado DES. Las permutaciones y las substituciones del algoritmo se desarrollaron en hardware ya que estas transformaciones representan las operaciones de mayor tiempo en la implementación en software del algoritmo DES. Al integrar las unidades funcionales realizadas en un procesador NIOS II como lógica adicional a la ALU se extiende el conjunto básico de instrucciones con lo cual se logra reducir el tiempo de ejecución para cada operación a tan solo uno o dos ciclos de reloj. Este enfoque permite combinar la flexibilidad que ofrece el software con el desempeño del hardware dedicado. Es importante destacar que la metodología seguida se puede extender a otros algoritmos de cifrado.

Referencias

[1] National Institute of Standards and Technology (NIST), FIPS PUB 46-3: Data Encryption Standard (DES), 1999.

[2] Altera, "NIOS II Custom Instructions User Guide", May 2008.

[21] R. B. Lee, Z. Shi, X. Yang., "Efficient Permutation Instructions for Fast Software Crytography", *IEEE Micro*, Nov/Dec 2001.