ANÁLISIS COMPARATIVO DE LOS SISTEMAS DE CIFRADO A5/1, W7 Y HÉLIX

DEIBY YESID CAÑÓN MERCHÁN

FUNDACIÓN UNIVERSITARIA SAN MARTÍN
FACULTAD DE INGENIERÍA
PROGRAMA DE INGENIERÍA EN TELECOMUNICACIONES
BOGOTÁ
2013 II

ANÁLISIS COMPARATIVO DE LOS SISTEMAS DE CIFRADO A5/1, W7 Y HÉLIX

DEIBY YESID CAÑÓN MERCHÁN
082040
DC082040@INGENIERIA.SANMARTIN.EDU.CO

MONOGRAFÍA

ASESOR TÉCNICO
HANS IGOR LÓPEZ CHÁVEZ

FUNDACIÓN UNIVERSITARIA SAN MARTÍN
FACULTAD DE INGENIERÍA
PROGRAMA DE INGENIERÍA EN TELECOMUNICACIONES
BOGOTÁ
2013 II

Nota de aceptación

_____
_____
_____
_____
_____
_____

_____
Hans Igor López Chávez
Asesor

_____
Carlos René Suárez Suárez
Jurado 1

_____
Natalia Isabel Fandiño Novoa
Jurado 2

Bogotá. 27/01/14

A mi madre Claudia Cañón y mi abuela María Merchán por su confianza, solidaridad y completo respaldo para culminar esta meta de mi vida.

A Karen Cabarcas, que considero como mi mentor en el ámbito académico, que el logro de esta meta le alegre enormemente y sea de respuesta al arduo trabajo que hizo educándome.

A mis queridos tíos Wilson, John y Edwin, por sus palabras de aliento y asistencia cuando siempre la requerí.

A mis amigos.

A pesar de las pocas expectativas que alguna vez la vida parecía ofrecer, el camino recorrido hasta el momento es grato y doy a gracias a Dios por poner en el camino personas con verdadera calidad humana, dedicados a formar talento humano antes que todo, ejemplos de responsabilidad y de compromiso social, pues es por ellos que esta meta es posible.

Gracias totales…

AGRADECIMIENTOS

A Dios por darme fuerzas para culminar esta meta, por poner en el camino excelentes docentes, compañeros de estudio, amigos y por el respaldo de mi familia.

A mi madre y abuela por su respaldo total a lo largo de la carrera.

Agradezco a mi compañero de promoción Luis Fonseca.

El más sincero agradecimiento a aquellas personas que hicieron este proyecto posible, al ingeniero Jorge Arévalo, Hans López y Juan Rodríguez, por su disponibilidad, por su ayuda y atención incondicional.

CONTENIDO

LISTA DE TABLAS

pág.

LISTA DE FIGURAS

pág.

LISTA DE ECUACIONES

pág.

LISTA DE ANEXOS

RESUMEN

Desde el surgimiento de los sistemas digitales, en especial de las redes telemáticas, la información ha cobrado más importancia que nunca, se ha vuelto un problema a nivel mundial, ya que estas redes por donde viaja la información son compartidas y cubren prácticamente el globo. Por eso se ha vuelto un requisito fundamental, tanto en el almacenamiento y como en el transporte brindar seguridad a la información.

El avance en criptología es enorme, se ha vuelto una ciencia formal, debido al profundo requerimiento del uso de las matemáticas y de otras áreas para las que aplica como sistemas digitales, sistemas de transmisión digital, redes, entre otras. Gracias a este avance hoy en día se dispone con buen catálogo de cifradores como eSTREAM, ECRYPT I, ECRYPTII, etc.

Implementar sistemas de cifrado brinda seguridad a la información, y realizar una respectiva comparación entre estos permite sugerir su aplicabilidad a distintas áreas y/o entornos de implementación.

INTRODUCCIÓN

En este proyecto se realizó el diseño e implementación de tres sistemas de cifrado de flujo en VHDL y se realizó una comparación basado en las simulaciones y en el resultado de compilación del software Quartus II Web edition. Los algoritmos de cifrado son A5/1, W7 y Hélix.

Primero en este trabajo se establece un marco referencial que trata sobre la seguridad de la información, los sistemas de cifrado, los algoritmos de cifrado selectos, el lenguaje de programación VHDL y los dispositivos FPGA, enfocándose a la familia de altera y sobre todo a la referencia a la que se sintetiza.

Luego de esta definición del marco referencial, se describen las características, ventajas y desventajas de los algoritmos de cifrado A5/1, W7 y Hélix.

Después se realiza el diseño de las entidades en VHDL para definir cada uno de los sistemas cifradores, una vez que se tiene diseñado cada una de las entidades, se ensamblan y se realiza una simulación funcional con vectores de prueba de los algoritmos W7 y Hélix. Para el caso de A5/1, se hace uso de una hoja de cálculo de Excel para comprobar la lógica del mismo.

Une ves comprobado el correcto funcionamiento de cada uno de los cifradores, se reúne la información entregada por el compilador para observar la cantidad de elementos lógicos de la FPGA que se están utilizando y la potencia que está siendo disipada debido al cifrador implementado.

Con el simulador de Quartus II, se realiza una simulación de tiempo, tomando una velocidad inicial y se va reduciendo hasta observar a qué velocidad los datos dejan de ser verídicos.

A partir de esta comparación de variables y de la implementación de los cifradores se concluye que los cifradores A5/1 y W7 son candidatos más próximos a una implementación de hardware por hacer uso de operaciones AND, XOR y de desplazamientos, además, estos cifradores operan independientemente del texto, favorable en una transmisión por capa física donde la pérdida de un dato no influirá sobre los demás. A pesar que estos dos cifradores posean estas mismas características, W7 es ocho veces más rápido que A5/1 y según su autor, es más seguro.

1. PROBLEMA

La gran cantidad de información digital que manipulan los bancos, las empresas, las organizaciones gubernamentales, entre otras, es afectada o es vulnerable sino se tiene un sistema de seguridad que mitigue el riesgo a amenazas, pues es blanco como por ejemplo de robo en cuentas bancarias o manejo inapropiado de dinero digital, si bien una violación en su integridad repercutirá además sobre terceros como usuarios e interesados comunes, trayendo una cadenas de eventos perjudiciales como perdida de dineros y de propiedad intelectual. Esto en parte se debe al fraude informático. En el 2012 en una encuesta por Kroll a nivel mundial realizada a más de 830 personas ejecutivas de distintas industrias y funcionarios, se reportó que el 23% de ellas se vieron afectadas por el robo informático, un 30% de ellas se sienten altamente vulnerables por este y un 55% de ellas tienen controles de fraude en ejecución (Kroll, 2013). Debido a estas estadísticas, se debe y se necesita implementar las herramientas necesarias para evitar y mitigar en lo posible el fraude informático, como sistemas de cifrado o cifradores, procedimientos de archivo, privilegios, permisos, etc.

Los algoritmos y protocolos de cifrado modernos son aplicados para brindar seguridad a la información digital (Goldwasser & Bellare, 2008). Las implementaciones de cifradores se hacen tanto a nivel de software como hardware según donde se desee implementar, además, se destacan ciertas características en cuanto a seguridad y ejecución de acuerdo al tipo de implementación. Por ejemplo a nivel de software resulta ser rentable en aplicaciones pequeñas (Hietala, 2007), pero presenta mayor debilidad ante distintos ataques como códigos maliciosos, ataques de fuerza bruta, ataques cold-boot, entre otros característicos de este nivel (Kingston, 2013). A demas, se generan cuellos de botella en el procesador del anfitrión por la petición de recursos a este (VIA, 2013). A parte de tener una buena cantidad en formas de ataque, es ineficiente al reducir el desempeño.

Centrando la importancia del problema sobre las telecomunicaciones, sin prestar atención a las aplicaciones que se estén atendiendo, y enfocándose únicamente en la comunicación, lo primordial sería la creación de un canal seguro que se daría sobre la capa física según el modelo de descripción de red OSI, por lo que se solucionaría con implementaciones en hardware, pero antes de proceder, habría que considerar otras variables como el ruido del medio, la longitud de los datos y otras variables que se deban atender para garantizar la comunicación (Douglen, 2010) y partiendo de ahí, variables de interés como costo de memoria, velocidad, consumo de potencia eléctrica, etc. Por medio del estudio de la criptografía y de otras ciencias donde esta se ha visto involucrada, se puede identificar parte de las variables y características de los cifradores y sus implementaciones, existen variables que no se pueden evaluar, que están por fuera del algoritmo o protocolo de cifrado, como la velocidad, consumo de potencia eléctrica, la memoria requerida para el funcionamiento del sistema cifrador / descifrador, y otras variables

dependientes al hardware y a los programas de implementación, en este caso Quartus II Web Edition.

De los tantos algoritmos disponibles como los de eSTREAM, NESSIE, ECRYPT y de distintos proyectos que se pueden implementar, la pregunta sería cuál es más apropiado para implementar, que brinde seguridad (cifrado de datos) y sea acorde a las necesidades para las cuales se solicita el hardware, por ejemplo si el hardware fuera requerido para dispositivos móviles, la prioridad sería el menor consumo de potencia, otros requerimientos como velocidad, cantidad de memoria requerida, costo del hardware, etc. Pero estas variables mencionadas, no pueden ser medidas simplemente al estudiar el algoritmo o protocolo de cifrado detenidamente, que puede dar una idea por ejemplo de cuanta memoria requeriría, pero no es un equivalente a una implementación realizada mediante herramientas de software, pues estas herramientas sintetizan y optimizan en los dispositivos programables de forma independiente, solo basándose en el código de descripción de hardware VHDL.

Un análisis comparativo de cifradores de flujo implementados en una FPGA, sirve de prototipo de hardware, que puede dar respuesta a la selección de algoritmos a desempeñarse en la capa física según los requerimientos. A partir de lo descrito anteriormente y con base al planteamiento de hacer una comparación, se genera la siguiente pregunta:

¿Cuál es el desempeño de los algoritmos, comparando el consumo de potencia, uso de recursos y síntesis en VHDL mediante su implementación en la FPGA Altera EP2C20F484C7?

2. JUSTIFICACIÓN

Implementar cifradores de flujo es una de las mejores formas de brindar seguridad (cifrado de datos), se ha escogido los cifradores de flujo A5/1, W7 y Hélix porque son una aproximación al cifrado One-Time-Pad, el cual es matemáticamente inquebrantable o perfectamente seguro si es implementado adecuadamente (Protechnix, 2013). Su implementación en hardware es simple, ya que el cifrado se realiza bit a bit, esto además es muy ventajoso en aquellos casos donde se desconoce la longitud de los datos. (Abd Al-Rasedy & Al-Swidi, 2011). Cabe también resaltar que la investigación sobre cifradores de bloque ha presentado mayor progreso que sobre los cifradores de flujo, haciendo más difícil tener un curso de criptoanálisis de cifradores de flujo que se base en artículos académicos (Schneier, 1999), por lo que se considera para tener un punto adicional en seguridad.

El proyecto se realiza por varios motivos, principalmente por observar el desempeño de distintos cifradores a nivel de hardware, y de ahí, escoger el que más se adecua a una aplicación puntual, como por ejemplo, se considera velocidad, memoria, consumo de potencia eléctrica, etc. La observación o más específicamente el análisis comparativo se realiza mediante la implementación de los algoritmos de cifrado selectos en la FPGA disponible en la Fundación Universitaria San Martín (FUSM) haciendo uso del lenguaje de descripción de hardware VHDL y de las herramientas incluidas en el software de Altera "Quartus II Web Edition". A pesar de que se pueda elaborar un circuito integrado de un sistema cifrador/descifrador basado en lenguaje de código en VHDL, resulta muy costoso si se tratase de unos cuantos circuitos y de cuidado ya que no se pueden realizar cambios sobre los circuitos de elaboración manual, en cambio, hacer uso de dispositivos reprogramables como las FPGA, sirven como prototipo de hardware, son asequibles, vienen en distintas gamas para satisfacer los requerimientos de los desarrolladores, y por supuesto, al ser reprogramables se puede realizar cambios cuando se necesiten.

Hacer cifrado y descifrado de forma rápida sobre medios físicos mediante un dispositivo electrónico da gran margen de probabilidad en el que se puede dar protección y seguridad de la información (cifrado de datos) al crear un canal seguro. Además, se puede implementar sobre modelos de redes que siempre dependen de la capa física, como el modelo OSI, y brindar un nivel de seguridad mayor.

Hoy en día se manejan sistemas criptográficos de seguridad fiables (llaves públicas, firmas digitales, certificados, etc.) al mismo tiempo o en un solo canal para su transmisión, esto hace que la tarea del criptoanalista se dificulta, así como existe un número determinado de formas para brindar seguridad, así lo hay en formas de ataque (Schneier, 1999).

Contar con un dispositivo electrónico de este tipo en la universidad, será de utilidad para interesados en el área de la electrónica digital y las telecomunicaciones, pudiendo así, entender la importancia de la seguridad en la información mediante prácticas de laboratorio, y puedan observar los avances que se han hecho sobre el tema y la importancia que tiene hoy en día.

3. OBJETIVOS

3.1 OBJETIVO GENERAL

Realizar un análisis comparativo de tres cifradores de flujo sintetizables para la FPGA Altera EP2C20F484C7.

3.2 OBJETIVOS ESPECIFICOS

Describir las características, ventajas y desventajas de los algoritmos de cifrado a implementar: A5/1, W7 y HELIX.

Diseñar las entidades en VHDL que soportan la lógica de cada uno de los cifradores y descifradores.

Realizar la descripción en hardware (VHDL) de los sistemas de cifrado A5/1, W7 y HELIX, y sintetizarlos para la FPGA Altera EP2C20F484C7.

Simular los sistemas descritos para cifrado y descifrado de datos en el software gratuito de Altera, Quartus II Web Edition.

Sintetizar los sistemas descritos para la FPGA Altera EP2C20F484C7.

4. MARCO REFERENCIAL

4.1 MARCO CONCEPTUAL

Este marco conceptual reúne la información necesaria para poder comprender que son los sistemas cifradores, se abarca primero haciendo introducción sobre seguridad de la información que cabe mencionar, es una área de estudio bastante amplia al implicar distintas ramas de la ingeniería y la rama de las matemáticas. Posterior a esto se enfoca a la definición del cifrador, sus clases y los distintos tipos de algoritmos de cifrado.

4.1.1 Seguridad de la información

Gracias a la necesidad de brindar seguridad a la información, y a la evolución de los sistemas que la manejan, se da paso a la criptología, ciencia del estudio de lo secreto u oculto. Esta se divide en disciplinas de estudio. Empezando por la criptografía esta es la ciencia de la escritura oculta, del griego Kryptos “secreto, oculto” y grafía “escritura”, que reúne un conjunto de ciencias como las matemáticas, las ingenierías, entre otras para garantizar la seguridad de la información. En contraparte a la anterior está el criptoanálisis que busca como tener acceso a la información no autorizada, mediante métodos analíticos. Otras dos disciplinas semejantes a las dos anteriores, y que usualmente se les confunde son la esteganografía que trata de esconder los mensajes en otros medios de información comunes, dándole a los terceros la percepción de que son otros mensajes comunes y corrientes, y el estegoanálisis que trata de encontrar información oculta dentro de un mensaje común y corriente mediante el uso de técnicas (Arévalo Aldana, González, & Sánchez, 2009).

La seguridad de la información se traduce básicamente en hacer comprensible y autenticable la información únicamente a las entidades autorizadas. Pero entrando en detalle algunos de sus objetivos son (Alfred J. Menezes, 1996):

Privacidad o confidencialidad: Mantener secreta la información de todos a excepción de aquellos autorizados a verla.

Integridad de datos: Asegurar que la información no haya sido alterada por medios no autorizados o desconocidos.

Identificación o autenticación de entidad: Corroborar la identidad de una entidad.

Autenticación de mensaje: Corroborar la fuente de la información.

Firma: Un medio para enlazar información a una entidad.

Autorización: Transporte a otra entidad, de sanción oficial a realizar o ser.

Validación: Medio para proporcionar oportunidad de autorización para usar o manipular información o recursos.

Control de acceso: Restricción de acceso a recursos a entidades privilegiadas.

Certificación: Aprobación de información mediante una entidad confiable.

Sellado de tiempo: Registra el tiempo de la creación o existencia de la información.

Testigo: Verificar la creación o existencia de la información mediante una entidad ajena a la de creación.

Recibido: Retroalimentación de que la información ha sido recibida.

Confirmación: Retroalimentación de que los servicios se proveyeron.

Propiedad: Los medios para proveer una entidad con derecho legal para usar o transferir un recurso a otros.

Anonimidad: Ocultar la identidad de una entidad envuelta en algún proceso.

Al no repudio: Prevenir la negación de compromisos o acciones.

Revocación: Retractación de certificación o autorización.

4.1.1 El cifrador y sus clases

El cifrado básicamente es la codificación de un mensaje para hacerlo seguro. Por facilidad al mensaje original se le conoce como texto plano, y al codificado como texto cifrado. Los cifradores se clasifican en la forma en que manipulan el texto plano para dar el texto cifrado (Alfred J. Menezes, 1996).

Cifradores por sustitución: Toma bloques de unidades de información a los cuales sustituye sus símbolos por otros símbolos (Alfred J. Menezes, 1996).

Cifradores monoalfabéticos: Cada símbolo de un texto se cifra siempre con un mismo símbolo, donde el cifrado de cada símbolo es independiente al texto (Alfred J. Menezes, 1996).

Cifradores polialfabéticos: Este consiste en tener varios alfabetos para el cifrado, donde cada símbolo será cifrado correspondiendo a un alfabeto específico, la selección de dicho alfabeto se hace por rotación (Alfred J. Menezes, 1996).

Cifradores por sustitución homofónica: Este consiste en tomar el texto y mirar la probabilidad de ocurrencia de cada símbolo, de esto, se genera tanto símbolos sean necesarios para cifrar el símbolo, con el propósito de que los símbolos del texto cifrado sean equiprobables (Alfred J. Menezes, 1996).

Cifrador de Vigenere y Beaunfort: Este cifrador está formado por una matriz de 26*26 (por las letras del alfabeto), donde la llave es un set de caracteres que no se repiten y que asignan a cada letra un equivalente según el alfabeto que corresponda (Mancilla Tello, 1995).

Cifrador de llave corrida: La seguridad de los métodos de substitución se ve generalmente incrementado con la longitud de la llave. En el cifrador de llave corrida, la llave tiene la misma longitud del texto en clave (Mancilla Tello, 1995).

Cifradores por substitución poligramática: Todos los métodos anteriores de substitución, cifran una letra del texto en claro a la vez. Por medio del cifrado de bloques de letras, los cifradores poligramáticos dificultan significativamente los métodos criptoanalíticos que se basan en una evaluación de las frecuencias relativas. El cifrado usa substitución por diagramas tomando bloques de letras (Mancilla Tello, 1995).

Cifradores de transposición: Es una clase de cifradores de llave simétrica, que simplemente consiste en permutar un símbolo dentro de un bloque.

Cifradores de producto: Son aquellos que combinan técnicas de substitución y transposición.

Cifradores exponenciales: El cifrado se hace basado en el cálculo de exponenciales sobre un campo finito (Mancilla Tello, 1995).

Cifradores de flujo: Estos son cifradores simples, pues la longitud de sus bloques son de una unidad. La transformación de cifrado puede cambiar por cada símbolo.

4.1.2 Algoritmos de cifrado

Los sistemas criptográficos se clasifican en dos grupos principales: Los sistemas de cifrado clásicos y modernos. Los clásicos son aquellos antes de la segunda guerra mundial, que eran máquinas de rotores y cuya información se podía cifrar y descifrar con papel y lápiz, pero con la partida de la era moderna volvió obsoleto estos sistemas clásicos gracias al poder computacional y al avance de las matemáticas.

Partiendo de lo anterior, este cambio abrupto permitió el desarrollo de nuevos sistemas criptográficos definiéndose en dos familias o tipos, de clase simétrica o asimétrica. Por lo que actualmente los algoritmos de cifrado trabajan a partir del tipo de claves que se manejan como privadas y públicas, y que dependiendo de estas se clasifica en alguna de las familias nombradas.

Sistemas Asimétricos: La clave o llave pública y la firma digital son las aplicaciones más comunes en la criptografía asimétrica, estos están basados en la teoría de números (Anderson, 2008).

La técnica de la criptografía asimétrica consiste hacer de la seguridad del cifrador dependendiente de la dificultad de resolver cierto problema matemático. Los dos problemas usados en casi todos los sistemas de campo son factorización y logaritmo discreto.

Sistemas Simétricos: Estos sistemas usan una misma clave tanto para el cifrado o descifrado, esto se le conoce también por criptografía convencional ó clásica o de clave secreta o simétrica (Anderson, 2008).

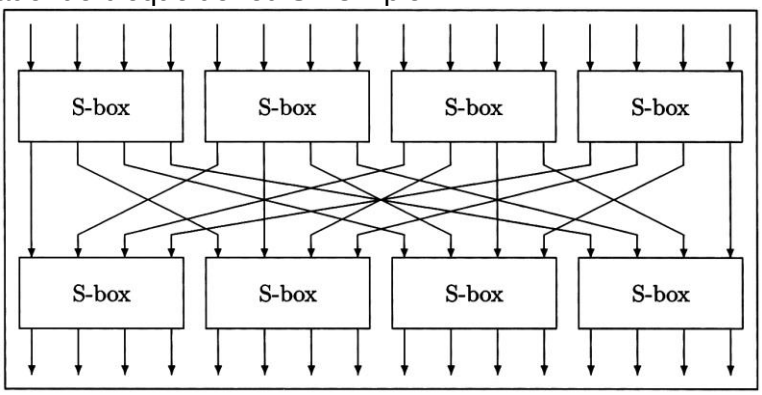
Claude Elwood Shannon propuso en los años cuarenta, que cifradores fuertes podían ser construidos al combinar substitución con transposición repetidamente. Por ejemplo, se agrega algún material de clave sobre un bloque de texto de entrada, después se alternan estos juegos de entrada, y se continúa de esta manera un número de veces. Shannon describió las propiedades de un cifrador de ser confuso y difuso al agregar valores de clave desconocidos que confundiría al atacante del valor de símbolo de un texto plano, mientras que la difusión hace referencia a distribuir el texto plano a través del texto cifrado (Anderson, 2008).

Cifrado simétrico de bloques: Los cifradores de bloque necesitan tanto confusión como difusión.

Los primeros cifradores de bloque consisten en simples redes que combinaban circuitos de substitución y permutación, por lo entonces llamados redes SP. El diagrama en la figura 1 muestra un red SP con 16 entradas, la cual se puede

imaginar cómo los bits de un numero de 16 bits, y dos capas de cajas invertibles de substitución de 4 bits (o cajas S), donde cada una puede ser visualizada como una tabla que contiene alguna permutación de los números del 0 al 15.

Figura 1 Un cifrador de bloque de red SP simple.



(Anderson, 2008)

El punto de este arreglo es hacer que si fuese a implementar una función arbitraria de 16 bits a 16 bits en lógica digital, se necesitaría 2^20 bits de memoria, una memoria de 2^16 bits para cada bit de salida. Esto requeriría de cientos de puertas lógicas, mientras que una función de 4bits a 4 bits toma tan solo 64 bits de memoria. Es de esperarse que con parámetros ajustables, la función producida por esta estructura pueda ser indistinguible de una función aleatoria de 16bits a 16 bits de un atacante que no conozca el valor de la clave. La clave debe consistir de alguna selección de un número de cajas S de 4 bits, o de ser agregada a cada ronda para proveer confusión, y el texto resultante alimentado a través de las cajas S para proveer difusión. El objetivo de lo anterior es utilizar un sistema de seguridad teniendo en cuenta tres aspectos necesarios para realizar un diseño seguro:

1. El cifrador necesita ser suficientemente "ancho"

2. El cifrador necesita tener rondas suficientes.

3. Las cajas S seleccionadas necesitan ser ajustables

4.1.3 Algoritmos de cifrado simétricos

Muchos de los algoritmos simétricos se basan en Feistel. Feistel es un método creado por Horst Feistel que usó el desarrollo de otros algoritmos como LUCIFER y DES. A continuación se describe y en la figura 2 y 3 se muestra su estructura.

- Un mensaje de una longitud N, típicamente 64 bits, 128 bits, etc., se divide en dos bloques de tamaño N/2, A y B.

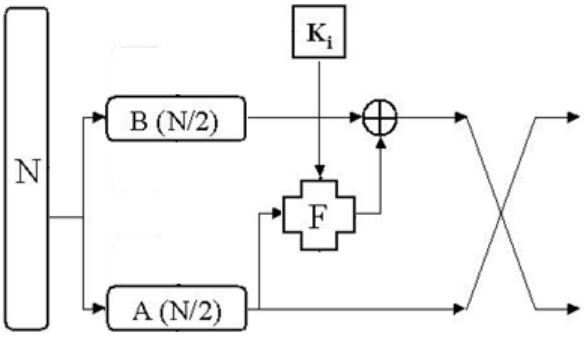
- Se toma una función F, cuya modificación sea bastante difícil.

- Se realizan una serie operaciones con la función F y la clave K_i con solo con una de mitades del mensaje A o B, cada iteración permuta la mitad. Este proceso es repetido n-veces.

En pocas palabras la red de Feistel es un método general, que transforma cualquier función dentro de una permutación.

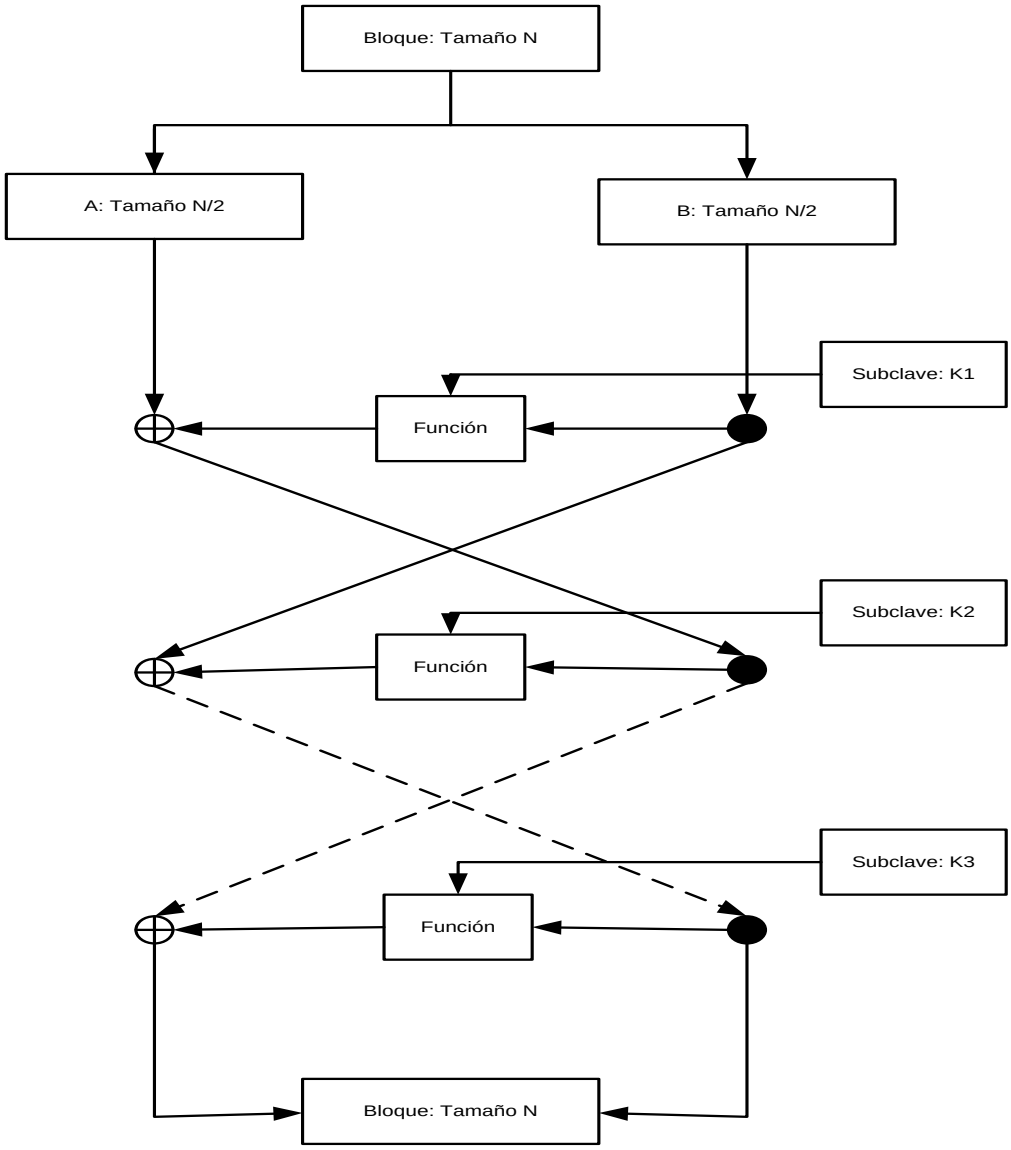
Este método de transformación es popular en los nuevos cifradores de bloques existentes como: FEAL, GOST, Khufu&Kafre, LOKI, CAST-128, Blowfish y RC5 (Barahona, 2001). Ver figura 2.

Figura 2. Red de Feistel.



(Alfred-λ, 2005)

Figura 3. Cifrado por bloque de Feistel.



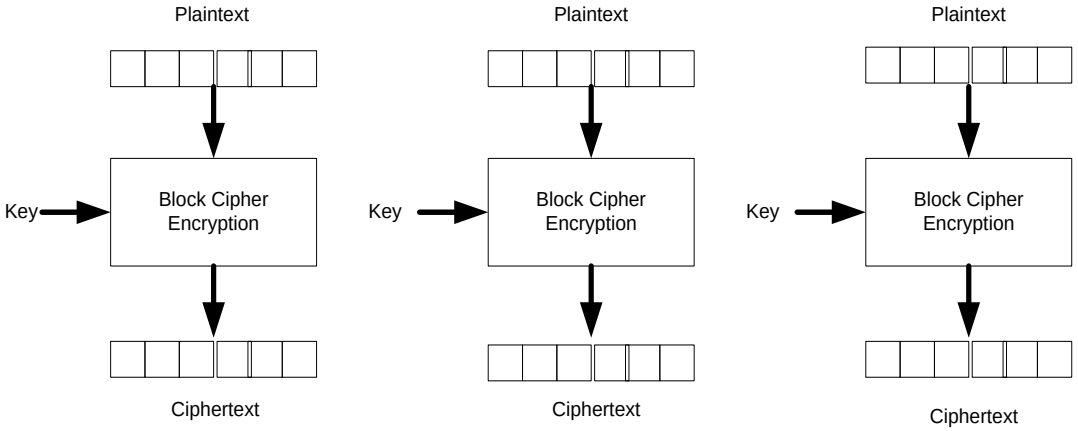
(Granados Paredes, 2006)

El modo de cifrado es necesario en un algoritmo tipo Feistel, pues el texto de entrada se debe descomponer en bloques de tamaño fijo, por ejemplo de 64 o 128 bits. Estos modos de cifrado se nombran a continuación:

- ECB (Electronic Code Book Mode)

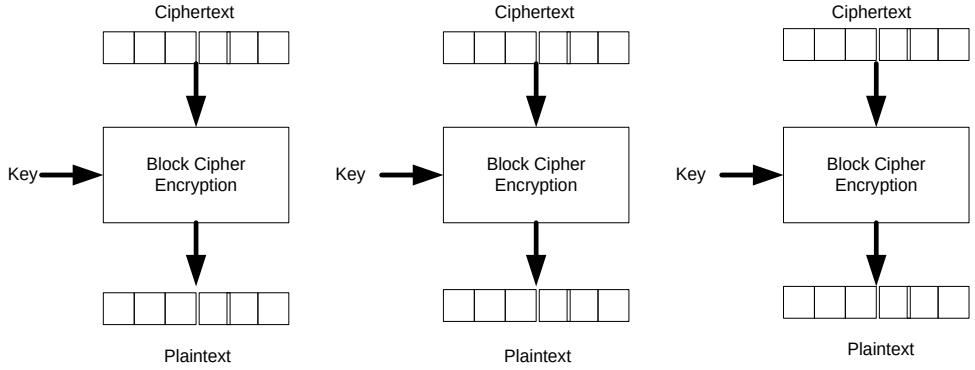
Este modo de cifrado estandarizado por el NIST (U.S. National Institute for Standards and Technology) es el más simple, pues parte el mensaje en bloques y se cifran por separado. En las figuras 4 y 5 se ilustra su funcionamiento.

Figura 4. ECB para el cifrado.



(Lerch, 2007)

Figura 5. ECB para el descifrado.

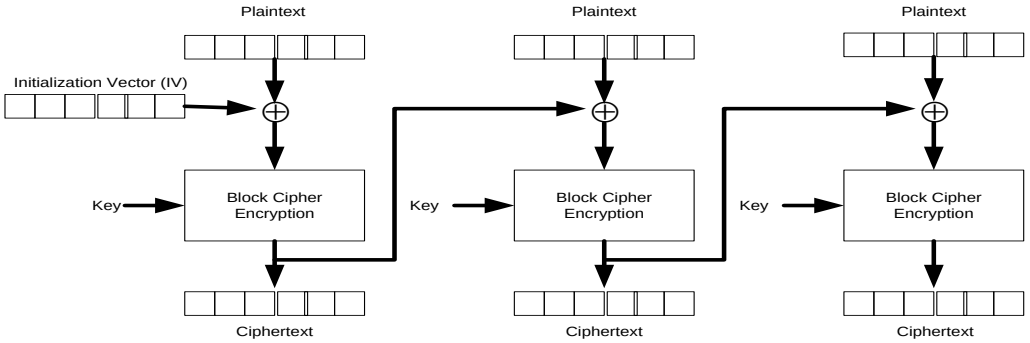


(Lerch, 2007)

- CBC (Cipher Block Chaining Mode)

Una extensión de ECB, la cual añade seguridad. Ha sido estandarizado por el NIST (U.S. National Institute for Standards and Technology). Se divide el mensaje en bloques, y se combina el cifrado del bloque anterior con el bloque de texto plano actual mediante una XOR. Pero como en el primer bloque no se cuenta con un bloque de texto cifrado para combinar, se hace uso de un vector de inicialización de suma importancia, pues si no se usa se es susceptible a ataques de diccionario, por lo consiguiente dicho vector debe ser aleatorio. En las figuras 6 y 7 se ilustra su funcionamiento.

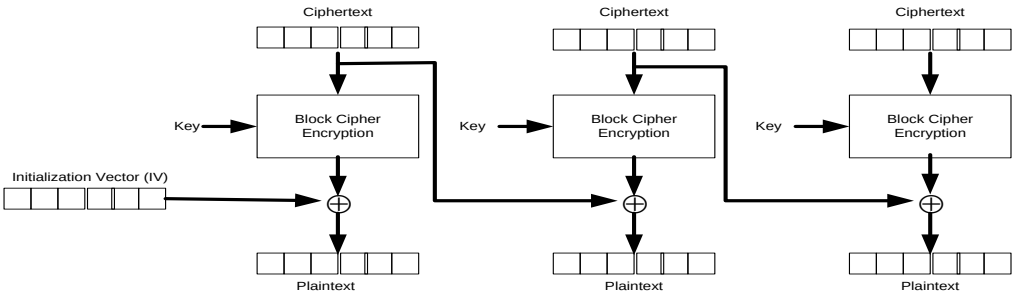
Figura 6. CBC para el cifrado.



(Lerch, 2007)

El mismo procedimiento pero inverso, descifra el mensaje:

Figura 7. CBC para el descifrado.

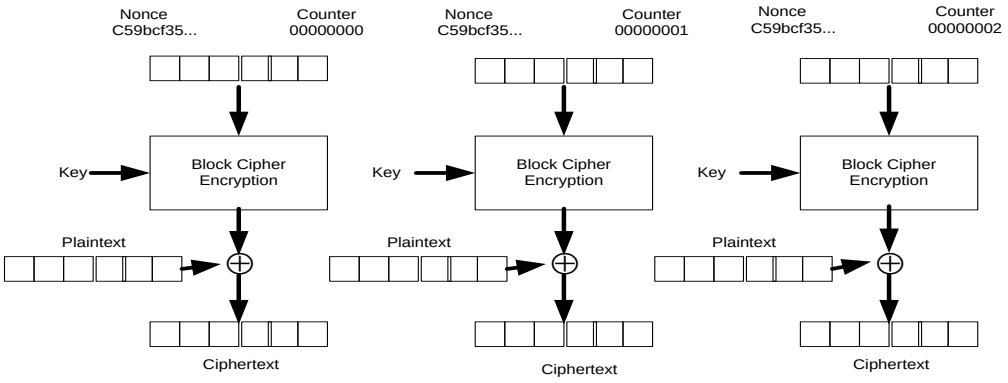


(Lerch, 2007)

- CTR (Counter Mode)

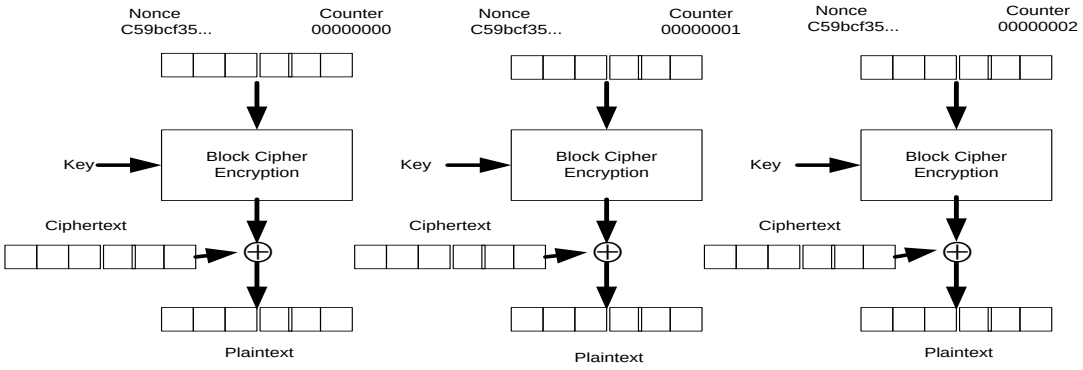
CTR simula un cifrado de flujo a diferencia de ECB y CBC, modos basados en bloques. Para simular dicho flujo, se produce una secuencia de cifrado, que es un flujo pseudo aleatorio. Para generar la secuencia de cifrado, se cifra un contador combinado con un número aleatorio (Nonce) mediante ECB y se va incrementando. Este flujo junto con el texto plano se combina mediante XOR para concluir el cifrado. Su funcionamiento se ilustra en las figuras 8 y 9.

Figura 8. CTR para el cifrado.



(Lerch, 2007)

Figura 9. CTR para el descifrado.

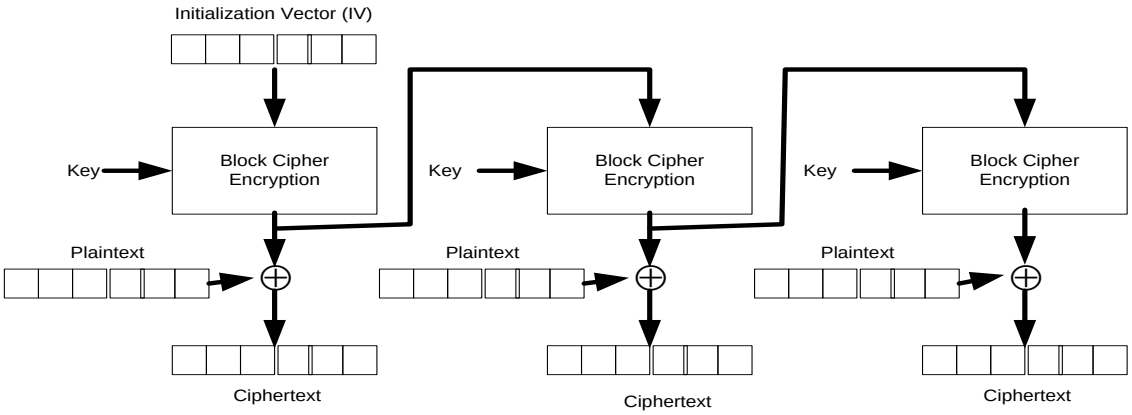


(Lerch, 2007)

- OFB (Output Feedback Mode)

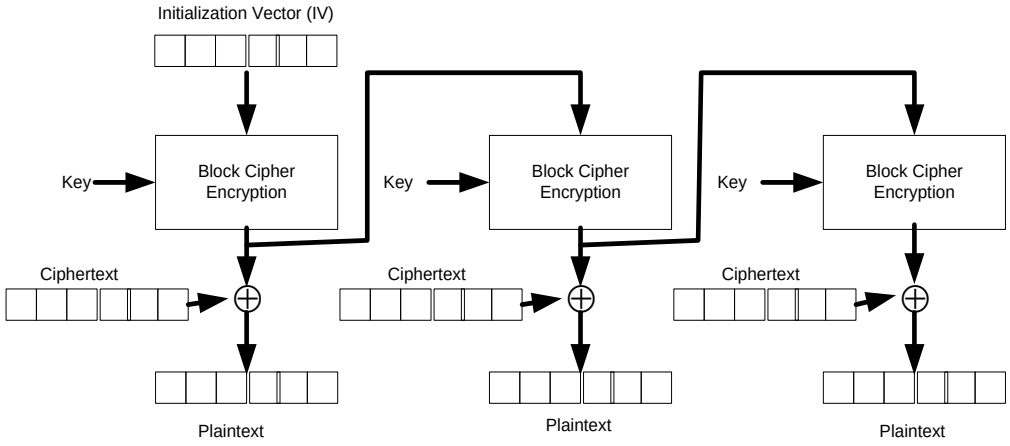
Como CTR es otro cifrado de flujo estandarizado por el NIST (U.S. National Institute for Standards and Technology). Cifrando el bloque de la secuencia de cifrado, resultante del cifrado del vector de inicialización, se genera otra secuencia. En las figuras 10 y 11 se ilustra su funcionamiento.

Figura 10. OFB para el cifrado.



(Lerch, 2007)

Figura 11. OFB para el descifrado.

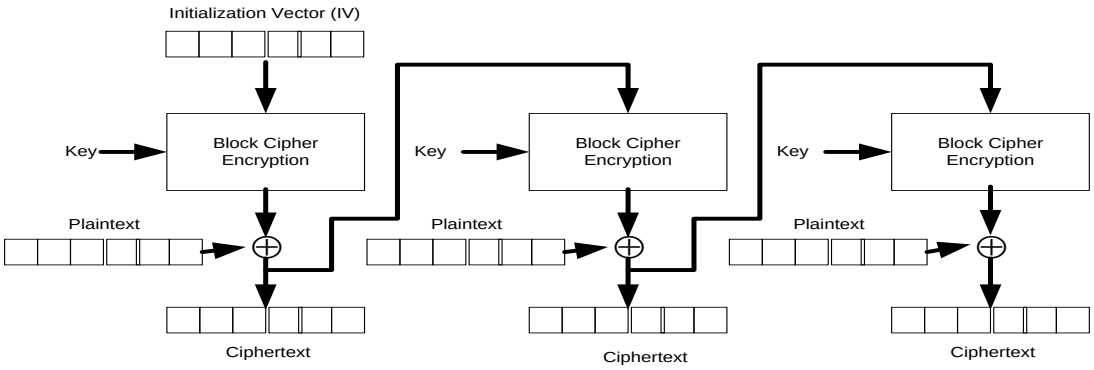


(Lerch, 2007)

- CFB (Cipher Feedback Mode)

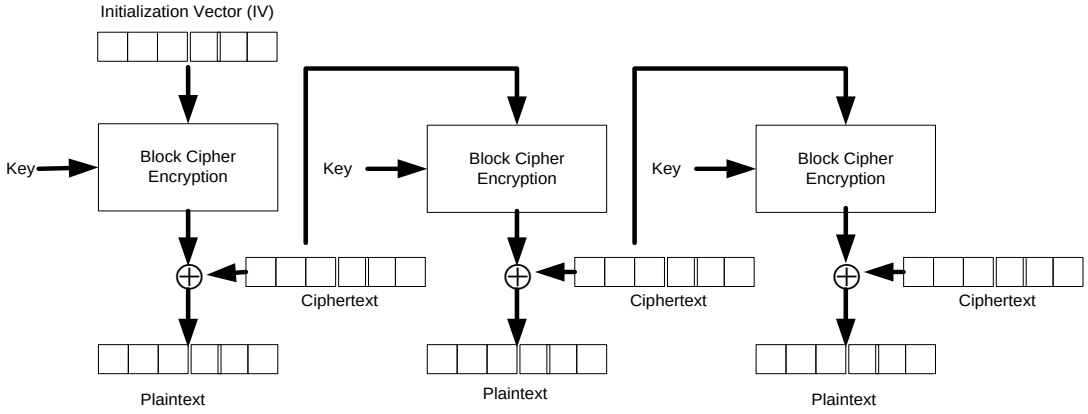
Muy similar a OFB, ha sido estandarizado por el NIST (U.S. National Institute for Standards and Technology). La secuencia de cifrado se produce al cifrar el último bloque de cifrado, en lugar del último bloque de la secuencia como lo sería en OFB. En las figuras 12 y 13 se ilustra su funcionamiento.

Figura 12. CFB para el cifrado.



(Lerch, 2007)

Figura 13. CFB para el descifrado.



(Lerch, 2007)

4.2 MARCO TEÓRICO

A5/1 es un algoritmo de cifrado utilizado en las redes GSM para proteger los datos de voz, su estado inicial es definido por una clave privada de 64 bits y por el número de trama de 22 bits, el cual es público, el cifrador cifra tramas de voz de 228 bits.

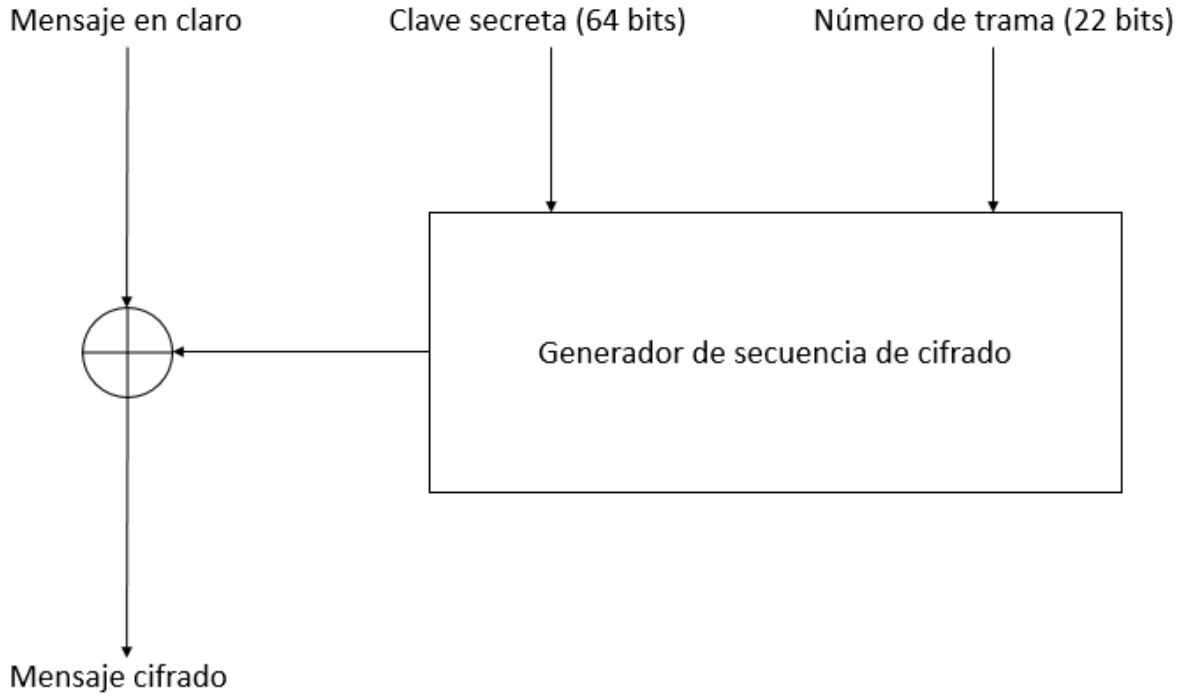
El algoritmo W7 es una propuesta en reemplazo a A5/1 por cuestiones de seguridad, su estado inicial depende de una clave privada de 128 bits y a diferencia de A5/1 que cifra bit por bit, este lo hace por bytes.

Hélix no tiene una aplicación específica. A diferencia de A5/1 y W7 el cifrado es dependiente del texto plano, su estado inicial depende de una clave privada variable de 0 a 256 bits y del nonce de 128 bits, el nonce es público y su nombre hace referencia a un número aleatorio generado por otra entidad.

4.2.1 Algoritmo A5/1

El algoritmo A5/1 es una versión del algoritmo A5 original, se opta por implementar este por ser una de sus versiones más fuertes. El algoritmo se ilustra en la figura 14.

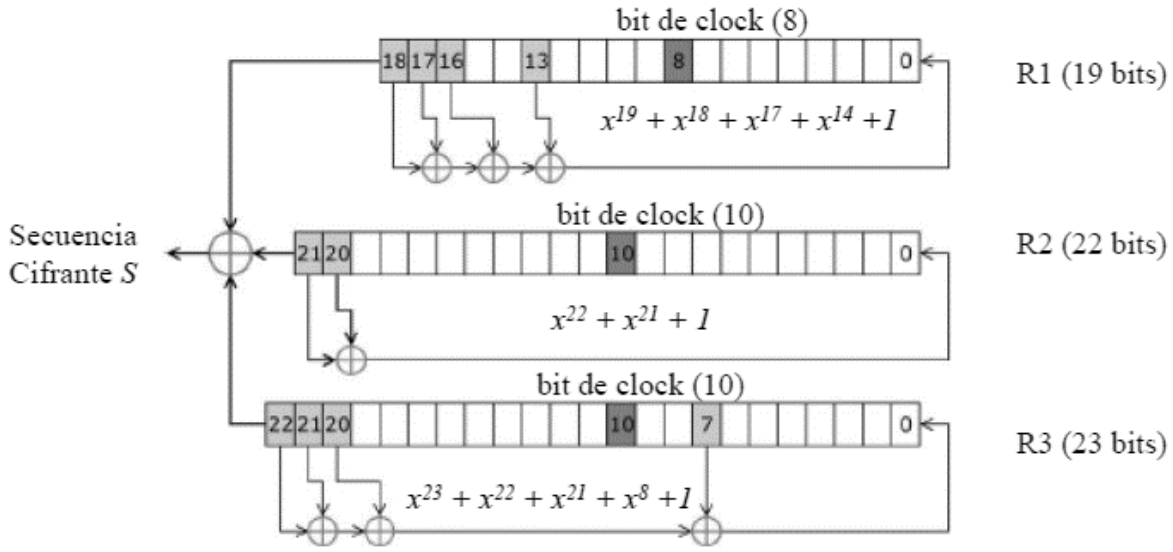
Figura 14. Cifrador A5/1.



(Saluja, 1991)

La secuencia de cifrado se genera a partir de 3 registros de desplazamiento realimentados linealmente o conocidos como registros LFSR. La importancia de estos registros, son su capacidad para generar vectores exhaustivos y/o aleatorios, su detección de errores como sus propiedades de corrección, por lo cual ha sido candidatos para su uso en sistemas cifradores. En este algoritmo la longitud de estos registros son de 19, 22 y 23 bits. En la figura 15 se ilustra el esquema del generador (Saluja, 1991).

Figura 15. Esquema de la secuencia de cifrado para el algoritmo A5/1.



(Saluja, 1991)

Como entradas para la generación de la secuencia de cifrado, utiliza la clave secreta y el número de trama. La clave secreta está constituida de 64 bits, a la igual suma los registros LFSR, y el número de trama de 22 bits, en total se usa 86 bits de entrada.

En los registros de la Figura 14 se observan posiciones de bits resaltadas que se operan con una XOR, es decir, hay una función por LFSR, las cuales son:

LFSR R1: $f_1(x) = x^{19} + x^5 + x^2 + x + 1$ genera $a = \{a(t)\}$

LFSR R2: $f_2(x) = x^{22} + x + 1$ genera $b = \{b(t)\}$

LFSR R3: $f_3(x) = x^{23} + x^{16} + x^2 + x + 1$ genera $c = \{c(t)\}$

Los bits de reloj especificados en la figura 14 son para la función mayoritaria, la cual se define como:

Ecuación 1. Función mayoritaria.

$$F(B_{R1}, B_{R2}, B_{R3}) = B_{R1}B_{R2} \oplus B_{R1}B_{R3} \oplus B_{R2}B_{R3}$$

(Saluja, 1991)

Las abreviaturas B_{R1}, B_{R2}, B_{R3} hacen referencia al valor del bit selecto como bit de reloj del registro.

El propósito de la función mayoritaria es indicar a los registros LFSR cuando hacer desplazamientos, estos desplazamientos se hacen en cada registro efectuado cuando el bit de reloj coincide con la función mayoritaria. La función fuerza al menos dos registros para que se desplacen.

Tabla 1. Función mayoritaria de A5/1 y W7

B_{R1}	B_{R2}	B_{R3}	F	Desplazamiento		
0	0	0	0	$R1$	$R2$	$R3$
0	0	1	0	$R1$	$R2$	
0	1	0	0	$R1$		$R3$
0	1	1	1		$R2$	$R3$
1	0	0	0		$R2$	$R3$
1	0	1	1	$R1$		$R3$
1	1	0	1	$R1$	$R2$	
1	1	1	1	$R1$	$R2$	$R3$

(Saluja, 1991)

El funcionamiento del algoritmo consta de los siguientes pasos:

Paso 1: Todos los registros están en ceros lógicos y la salida está deshabilitada.

Paso 2: Las funciones de los registros son operadas (XOR) paralelamente con la clave secreta, esta operación va llenando el registro por medio del desplazamiento, del LSB al MSB, esto se lleva por supuesto por 64 ciclos, que es la longitud de la clave secreta.

Paso 3: Se repite el paso anterior, pero esta vez con el número de trama, que equivaldría a 22 ciclos.

Paso 4: Los registros son desplazados 100 veces, atenidos a la función mayoritaria. Una vez terminado la fase de iniciación ha finalizado.

Paso 5: Se habilita la salida para la secuencia de cifrado, las salidas de los registros viene siendo el bit más significativo operados entre sí con una XOR, a la vez los registros obedecen a la función mayoritaria. La operación se realiza 228 veces.

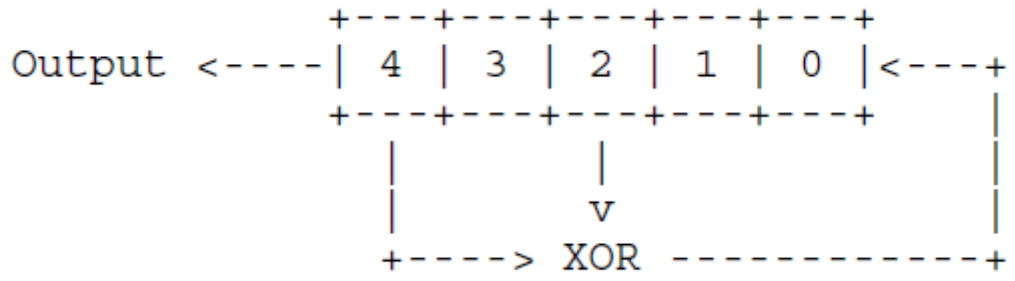
Paso 6: Con la secuencia de cifrado (228 bits) se opera (XOR) con la secuencia de texto en claro (228 bits).

4.2.2 Algoritmo W7

El cifrado W7 utiliza registros de desplazamiento de realimentación lineal (LFSR) como bloque de construcción básico. La siguiente figura muestra un LFSR sencillo, no un algoritmo actual. La figura 16 muestra un registro de desplazamiento de 5 bits. Los bits están numerados 4 a 0 de izquierda a derecha, y la salida se toma del cuarto bit (Thomas & Anthony, 2002).

En cada reloj, los bits de desplazamiento de derecha a izquierda, y el bit desplazado en el bit 0 es la OR exclusiva del bit 4 y el bit 2 (Thomas & Anthony, 2002).

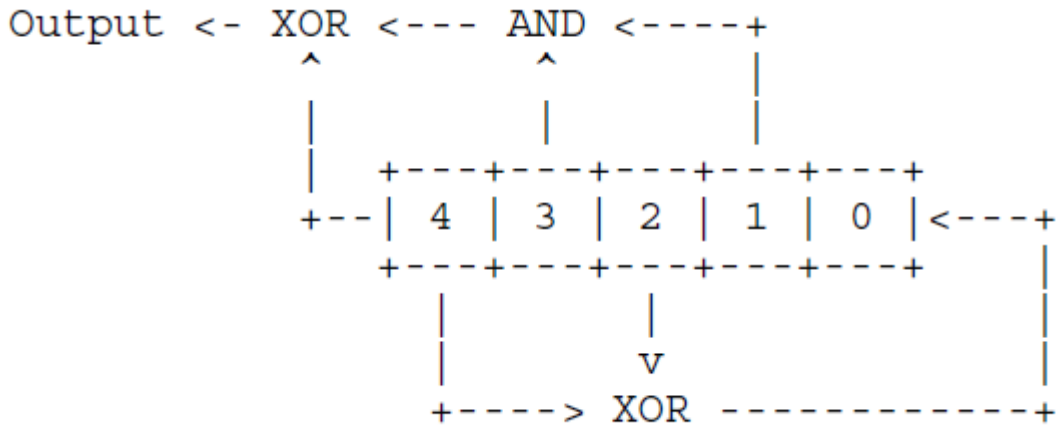
Figura 16. Registro de desplazamiento de realimentación lineal.



(Thomas & Anthony, 2002)

El cifrado W7 no utiliza el bit de salida directa. En su lugar, se utiliza una función de filtrado no lineal que es una combinación de varios bits en el registro de desplazamiento. La siguiente figura muestra un ejemplo sencillo de este enfoque. La salida real es la OR exclusiva de dos cantidades: (a) el registro de desplazamiento de salida y (b) la AND lógica de los bits de 3 y 1. (Thomas & Anthony, 2002)

Figura 17. LFSR lineal con función de salida.



(Thomas & Anthony, 2002)

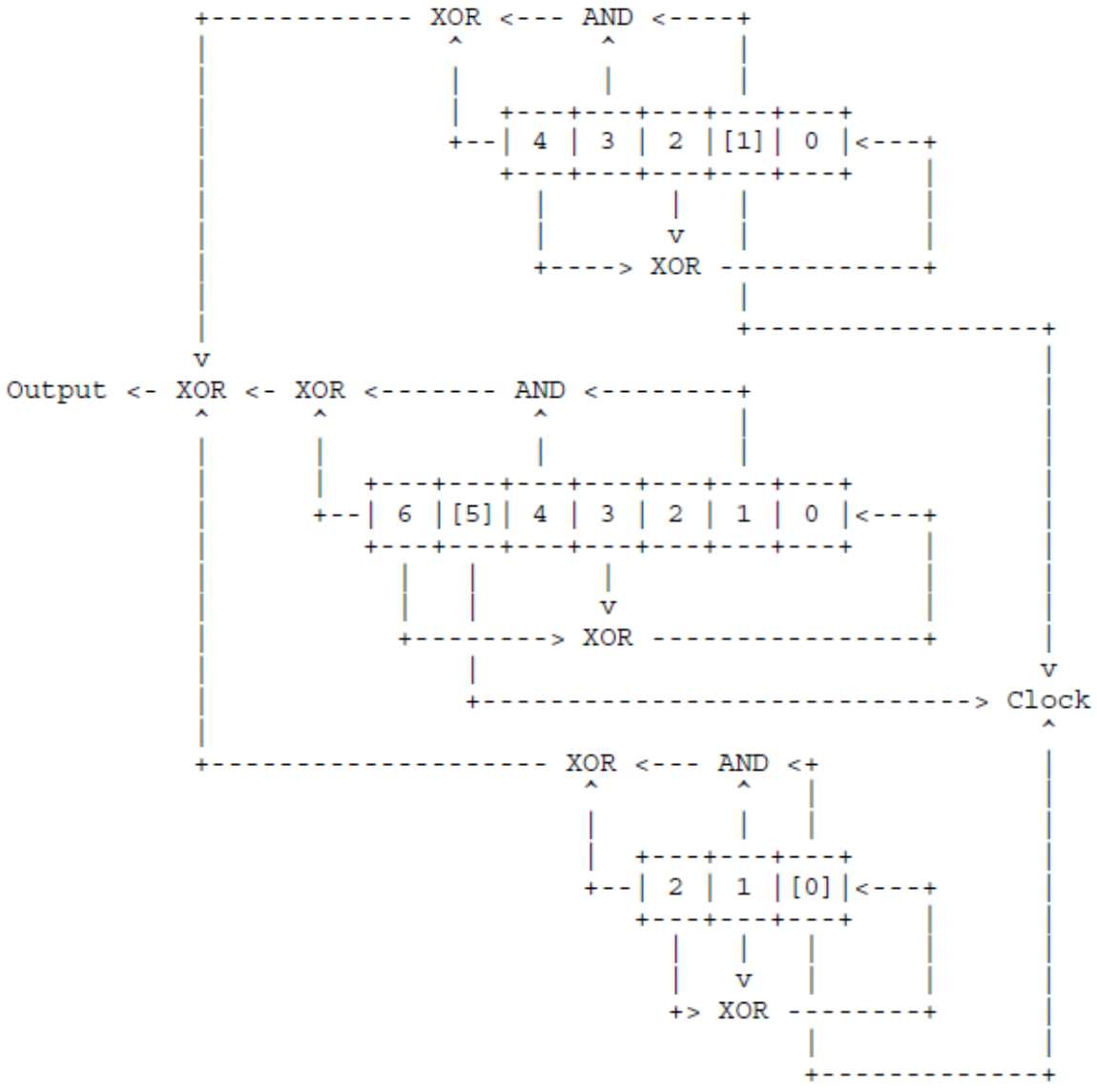
Para seguridad adicional, el sistema de cifrado W7 combina tres de estos LFSR de salidas filtradas. La salida real de las cadenas de claves se toma como la OR exclusiva de los tres LFSR individuales. Los tres LFSR juntos determinan también cuando cada registro debe desplazarse. Un bit en cada registro se designa como el habilitador de reloj para dicho registro. En cada ciclo de reloj, una función mayoritaria, evalúa los habilitadores y lo que sean mayoría en un valor, 0 o 1, se desplazarán, en la figura 18 se ilustra un ejemplo donde los habilitadores son los bits 1, 5 y 0 (Thomas & Anthony, 2002).

Los LFSR para cada una de las ocho secuencias son de 38, 43 y 47 bits de longitud. Su estado inicial, el cual es idéntico para todas las ocho secuencias, es la clave de cifrado simétrico. Los 128 bits de la clave se mapean en los LFSR de acuerdo a la siguiente tabla (Bit 0 es el menos significativo).

Tabla 2. Distribución de clave en la celda W7.

LFSRa (38 bits)	LFSRb (43 bits)	LFSRc (47 bits)
Bit 0 = Bit de clave 0 Bit 1 = Bit de clave 1 … Bit 37 = Bit de clave 37	Bit 0 = Bit de clave 38 Bit 1 = Bit de clave 39 … Bit 42 = Bit de clave 80	Bit 0 = Bit de clave 81 Bit 1 = Bit de clave 82 … Bit 37 = Bit de clave 127

Figura 18. LFSR regidos por un reloj irregular.

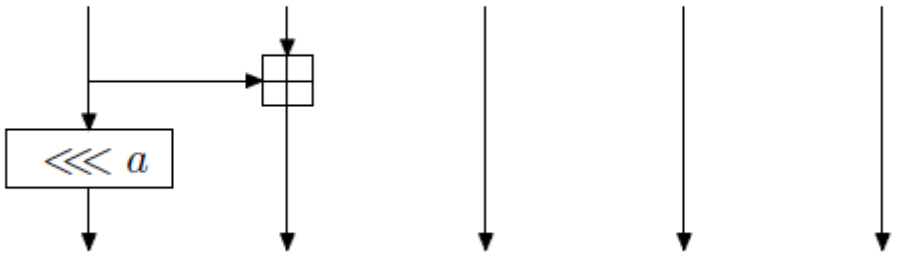


(Thomas & Anthony, 2002)

4.2.3 Algoritmo Hélix

Hélix es una combinación de un cifrado de flujo y una función MAC, y provee directamente la funcionalidad de cifrado autenticado. El diseño de seguridad de Hélix es de 128 bits, lo que significa que un ataque al sistema de cifrado requiere al menos 2^128 bloques funcionales de Hélix. Hélix puede procesar datos en menos de 7 ciclos de reloj por byte en una CPU de bajo rendimiento lo que lo convierte en un sistema de cifrado más rápido que AES. Hélix usa una clave privada de 256 bits y un nonce de 128 bits (Un número aleatorio de conocimiento público). Todas las operaciones son en palabras dobles (32 bits). Las únicas operaciones que se utilizan son adición módulo 2^32, OR exclusiva, y rotación en un número fijo de bits. La filosofía de diseño de Hélix se puede resumir como "muchas rondas simples". La ronda se ilustra en la figura 19 (Ferguson, y otros, 2003).

Figura 19. Ronda de Hélix.



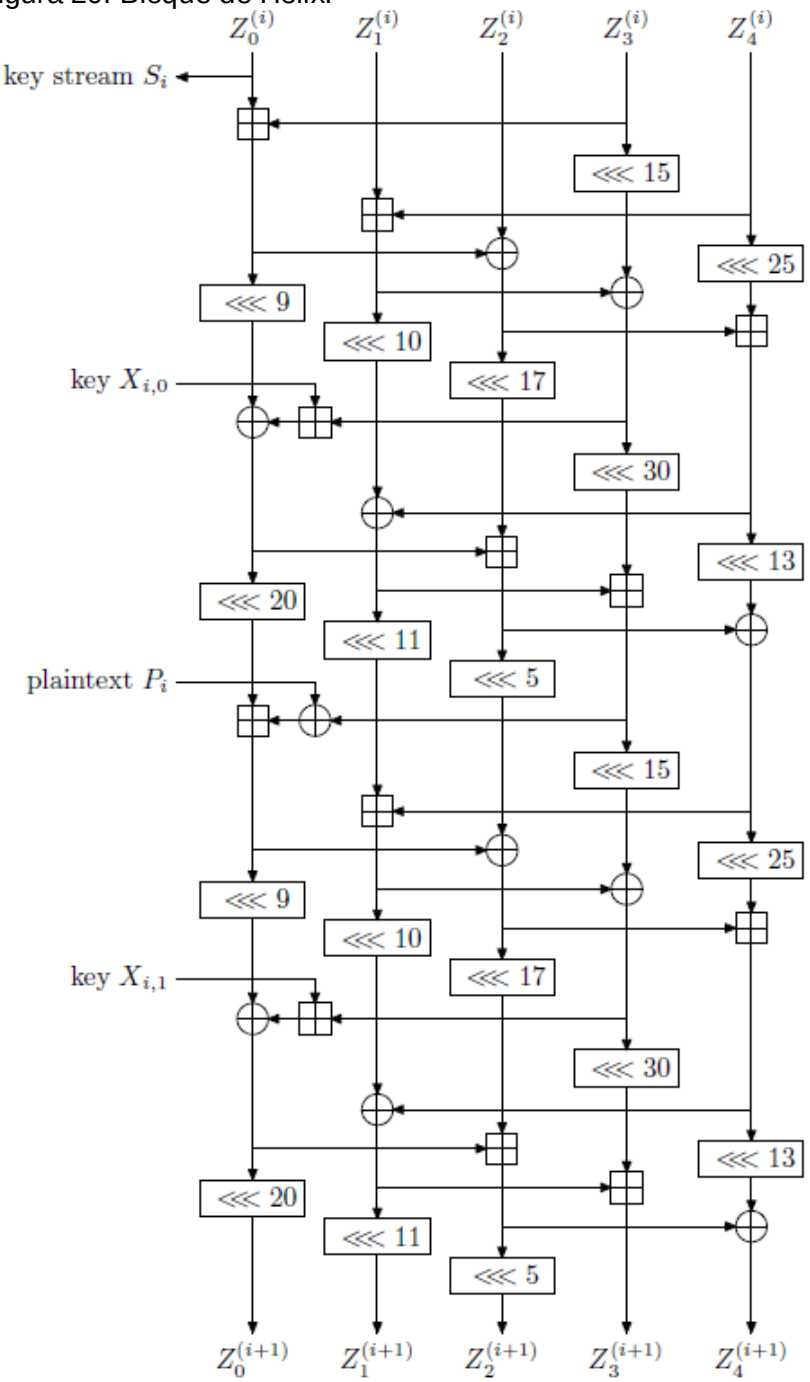
(Ferguson, y otros, 2003)

Hélix tiene un estado que consta de 5 palabras de 32 bits cada una. Una sola ronda de Hélix consiste en añadir aritméticamente o en OR exclusiva un estado de la palabra a la otra, y la rotación de la primera palabra. Esto se muestra en la figura 19, donde las palabras de estado se muestran como líneas verticales.

El bloque de Hélix, ilustrado en la figura 20, consta de veinte rondas que se aplica en forma cíclica a través de los estados de la palabra. Al principio del bloque, el estado $Z_0^{(i)}$ es la salida que cifra el texto plano, pero a la vez este mismo segmento de texto plano es introducido a la mitad del bloque para generar dependencia en el cifrado. En cuanto a las entradas $x_{i,0}$ y $x_{i,1}$ son palabras claves generadas a partir del nonce y la clave privada.

Se aplica cuantos bloques sean necesarios para cifrar el texto plano, pues cada bloque cifra 32 bits, la salida final del bloque alimentará la entrada del próximo bloque.

Figura 20. Bloque de Hélix.



(Ferguson, y otros, 2003)

La función de cifrado Hélix toma como entrada una clave de longitud variable U hasta de 32 bytes, un nonce N de 16 bytes, y un texto plano P. Se produce un mensaje de texto cifrado y una etiqueta que proporciona autenticación. La función de descifrado toma la clave, nonce, texto cifrado, y la etiqueta, y produce ya sea el mensaje de texto o un error si la autenticación no fue correcta. Hélix opera en palabras de 32 bits, mientras que las entradas y salidas son una secuencia de bytes (Ferguson, y otros, 2003).

En todas las situaciones Hélix utiliza la convención primer byte, byte menos significativo. Una secuencia de bytes x_i se identifica con una secuencia de palabras X_j por las relaciones:

Ecuación 2. Relación palabra bytes.

$$X_j := \sum_{k=0}^{3} x_{(4j+k)} \cdot 2^{8k}$$

Ecuación 3. Relación bytes palabra.

$$x_i := \left\lfloor \frac{X_{\lfloor i/4 \rfloor}}{2^{8(i \bmod 4)}} \right\rfloor \bmod 2^8$$

(Ferguson, y otros, 2003)

Si se denota $l(x)$ como la longitud de una cadena de bytes x, entonces la entrada de clave privada U consiste en una secuencia de bytes $U_0, U_1, U_2, ..., U_{l(U)-1}$ con la restricción de $0 \le l(U) \le 32$ bytes. La clave es procesada a través de la función de mezcla de clave.

La función de ronda F, necesaria para la función de mezcla de clave, mapea 128 bits a 128 bits. Esta función hace uso del bloque Hélix para el mapeo, introduce cuatro palabras y adiciona una demás con el valor $l(U)+64$ para completar las cinco palabras en el bloque, las demás entradas del bloque tienen valor cero. Las primeras cuatro palabras resultantes del bloque forman el resultado de F.

La función de mezcla de clave extiende la clave privada U con $32 - l(U)$ bytes inicializados en cero. Convierte los 32 bytes de la clave en ocho palabras $K_{32}, ..., K_{39}$. Las demás palabras de clave son definidas por la ecuación 4.

Ecuación 4. Función para establecer clave de trabajo.

$$(K_{4i},...,K_{4i+3}) := F((K_{4i+4},...,K_{4i+7})) \oplus (K_{4i+8},...,K_{4i+11})$$

$$i = 7,...,0$$

(Ferguson, y otros, 2003)

Lo anterior define la clave de trabajo que consiste de ocho palabras de $K_0,...,K_7$. El texto plano P y el texto cifrado C ambos son secuencias de bytes de la misma longitud, con la restricción $0 \le l(P) \le 2^{64}$. Ambas son manipuladas como una secuencia de palabras de P_i y C_i. La última palabra del texto plano y texto cifrado pueden ser parcialmente utilizados. Los bytes extra en el texto plano de la última palabra son tomados como cero. Los bytes extra en el texto cifrado son irrelevantes y nunca son utilizados.

Como se puede observar en la figura 20, el bloque mapea 160 bits a 160 bits teniendo como entrada las palabras claves y el texto plano. Las ecuaciones 2 y 3 son complementarias y muestran la conversión de bytes a palabras dobles y viceversa.

Para la generación de las palabras claves, se extiende el nonce a 8 palabras por medio de la ecuación 5 y las palabras claves se definen por la ecuación 6.

Ecuación 5. Extensión del nonce.

$$N_k := (k \bmod 4) - N_{k-4} (\bmod 2^{32})$$

$$k = 4,...,7.$$

Ecuación 6. Palabras claves.

$$X_{i,0} := K_{i,0} \bmod 8$$

$$X_{i,1} := K_{(i+4) \bmod 8} + N_{i \bmod 8} + X'_i + i + 8$$

$$X'_i := \begin{cases} \lfloor (i+8)/2^{31} \rfloor \Rightarrow i \bmod 4 = 3 \\ 4 \cdot l(u) \Rightarrow i \bmod 4 = 1 \\ 0 \end{cases}$$

El estado inicial del cifrador se define por la ecuación 7, donde el texto plano es cero y la secuencia de cifrado se descarta. Ocho bloques se aplican de -8 a -1.

Ecuación 7. Iniciación de Hélix.

$$Z_i^{(-8)} = K_{i+3} \oplus N_i \Rightarrow i = 0,...,3$$

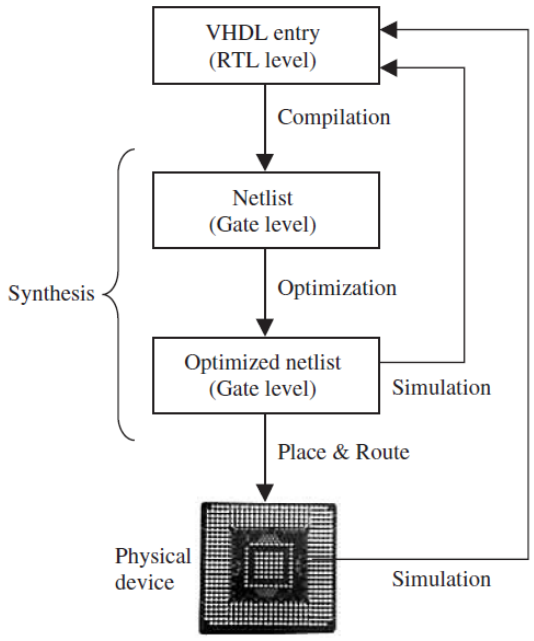
$$Z_4^{(-8)} = K_7$$

Concluida la iniciación se procede a cifrar el texto plano al aplicar un número K de bloques, donde $K := \lfloor (l(P)+3)/4 \rfloor$.

4.2.4 VHDL

VHDL es un lenguaje de descripción de hardware en donde se especifica las características lógicas requeridas para un comportamiento digital esperado. VHDL es un sinónimo de VHSIC que es la abreviación de circuitos integrados de alta capacidad (Very High Speed Integrated Circuits) iniciado en el departamento de defensa de Los Estados Unidos de América en la década de los 80 (Pedroni, 2004).VHDL a lo largo de los años se ha visto evolucionando desde su primer versión VHDL 87, el VHDL 93 y por último el VHDL 2008 estandarizado como un lenguaje de descripción de hardware IEEE 1076 pero tiene estándares adicionales como IEEE 1164 para introducir sistema lógico de valores múltiples.

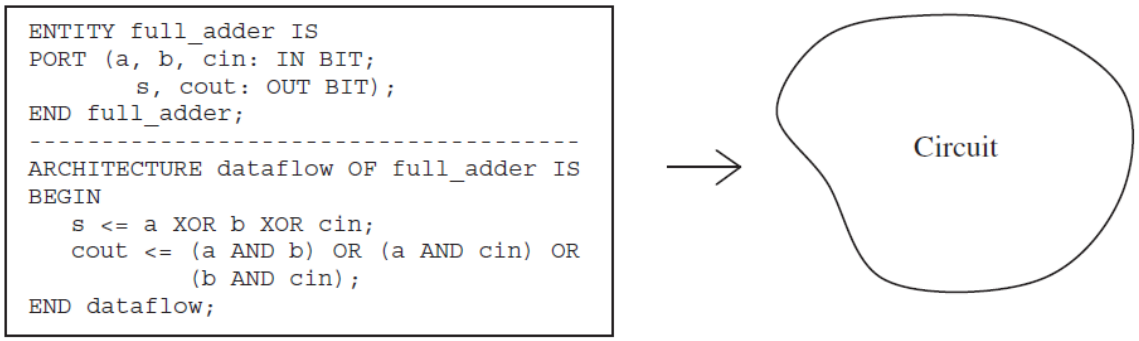
Figura 21. Secuencia de diseño de VHDL.



(Pedroni, 2004)

El proceso de crear un código VHDL inicia con la creación de una entidad que tiene como extensión .vhd y el nombre de la entidad anticipándolo. El primer paso en el proceso de síntesis es la compilación. Compilación es la conversión del lenguaje VHDL de alto nivel, donde se describe el circuito en el nivel de transferencia de registro (RTL), en una lista de conexiones en el nivel de compuertas. El segundo paso es la optimización, que se realiza en la lista de conexiones a nivel de compuertas para la velocidad o para el área. En esta etapa, el diseño puede ser simulado. Por último, un software de localización y enrutamiento (instalador) va a generar la disposición física para un chip PLD / FPGA o va a generar las máscaras para un ASIC (Pedroni, 2004).

Figura 22. Ejemplo de código VHDL.



(Pedroni, 2004)

Como se ve en la figura anterior, las variables son introducidas en la entidad y la parte sistemática es introducida en la arquitectura del código. El sistema describe como se hace una suma de dos bits con acarreo y un contador. La suma de bits está compuesta como $s = a \oplus b \oplus cin$ y el contador como $a \cdot b + a \cdot cin + b \cdot cin$.

4.2.5 Familia FPGA altera

ALTERA es un fabricante de instrumentación lógica fundada en 1983 en San José, California la cual es pionera en la lógica programable basada en algoritmos de sintonización por medio de microchips programables con memoria embebida y conmutadores lógicos inteligentes. Hoy en día cuenta con un portafolio de diseño general que proporciona versatilidad en cualquier campo de la electrónica. ALTERA divide sus dispositivos en familias relacionadas con diseño de fabricación tanto en las FPGAs, ASICs, CPLDs y otros dispositivos de categoría específica (por ejemplo equipos militares), en software tiene una gama de programas robustos en la configuración y simulación de circuitos digitales de complejidad como Quartus II y

Nios II (para la programación de procesadores embebidos), entre otros (Altera, 2013).

En FPGAs, ALTERA tiene una presentación de 3 familias con sus respectivas categorías:

Tabla 3. FPGAs de Altera.

Familia	Categorías
Stratix	GX ,II ,II GX, III (L y E) ,IV (E, GX, GT), V (E, GX, GS, GT)
Arria	GX, II (GX and GZ), V (GX, GT, GZ, SX, ST)
Cyclone	II, III (y LS), IV (E y GX), V (E, GX, GT, SE, SX, ST)

Los dispositivos Cyclone II tiene procesadores NIOS II que permiten soluciones integradas de ajuste personalizado. Los dispositivos Cyclone II poseen también gran variedad de periféricos, gran capacidad de memoria, gran variedad de entradas y salidas y un óptimo rendimiento de procesadores embebidos como lo son los NIOS II de tipo simple o múltiple para ser vinculados con otros procesadores o reemplazar en algunos casos. La unión de Cyclone II y Nios II permite soluciones de procesamiento integrado de alto desempeño y gran complejidad a bajo costo. Además de los componentes, los dispositivos Cyclone II permiten una amplia gama de interfaces y protocolos de entrada y salida utilizando la función Fast-on que ofrece un tiempo más rápido de entendimiento entre dispositivos (Altera, 2013).

4.3 ANTECEDENTES

El ser humano desde tiempos remotos ha notado la necesidad de una comunicación segura, de ser oculta para los demás por diversos objetivos. De los registros que se tienen, la necesidad de realizar una comunicación segura se estima en contextos bélicos, en civilizaciones antiguas como Egipto, la ordenes provenían de distintas jerarquías al mando, y la veracidad de la información era primordial, por lo tanto, habían asaltos o inclusive espías para interceptar ordenes e información importante, a consecuencia de estos eventos la criptografía nace como solución en privacidad y seguridad de la información.

La criptografía es el arte de la escritura secreta, pero se ha convertido formalmente en una ciencia, pues con el transcurso del tiempo ha recolectado un número de ciencias cuyo propósito es mantener la privacidad de la comunicación entre dos personas o entidades tomando el mensaje y codificándolo de tal forma que solo sea entendible para los destinatarios originales.

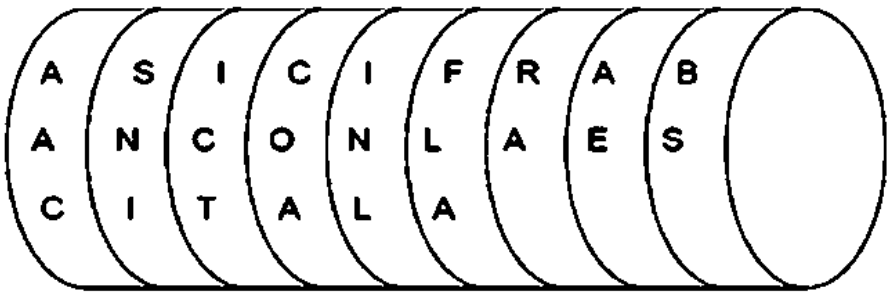
Cuando se habla de criptografía, se refiere a esconder, ocultar, hacer incoherente cierta información. La palabra criptografía viene del griego kryptus que significa escondido y de la palabra graphos que significa escritura.

Se suele confundir la criptografía con esteganografía, son distintos en cuanto cómo se trata los mensajes o la información. La esteganografía aplica un número de técnicas para ocultar los mensajes, por ejemplo tintas invisibles, inscripciones con símbolos, tablillas recubiertas de cera y se le atribuye a la imagen de los espías, pero a la final quedan obsoletos una ves son descubiertas.

Aplicada las técnicas de la criptografía se le asigna un nombre a los mensajes tratados, si el mensaje ha sido transformado se le llama cifrado y su respectiva inversa se llama descifrado. Para realizar este cifrado y descifrado se hace un número de procesos que obedece a unas reglas, a lo anterior se le resume y se llama clave, hay veces que también se le llama llave.

En el siglo V antes de Cristo se conoce un dispositivo llamado la escítala proveniente de los lacedemonios usado en la guerra entre Atenas y Esparta. Este consistía en un mensaje rellenado con símbolos innecesarios, y el mensaje podía ser visualizado al envolverse en un rodillo, por lo tanto la longitud del rodillo como su grosor eran prefijados. En la Figura 23 se ilustra la escítala.

Figura 23. Escítala con un mensaje oculto.

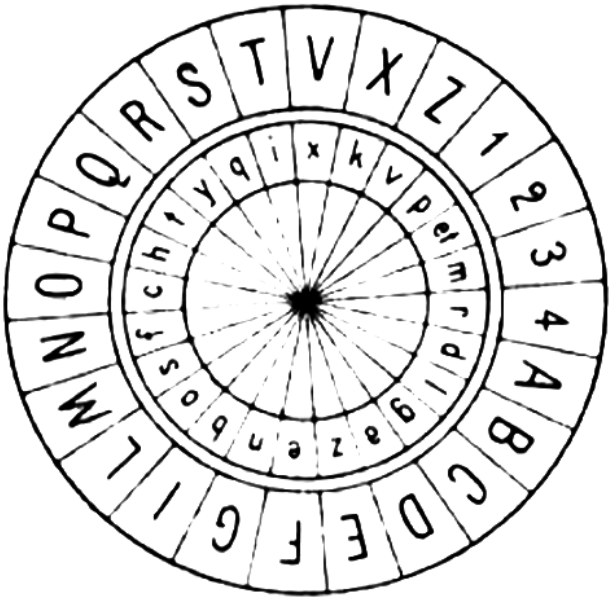


(Granados Paredes, 2006)

En el siglo XVI surgen nuevas máquinas como el disco de Alberti en 1466, un sistema polialfabético que contiene distintos abecedarios para realizar el cifrado del

mensaje. En este dispositivo tanto el emisor como receptor debían ponerse de acuerdo con la posición de los círculos concéntricos del disco. En la figura 24 se ilustra el mecanismo.

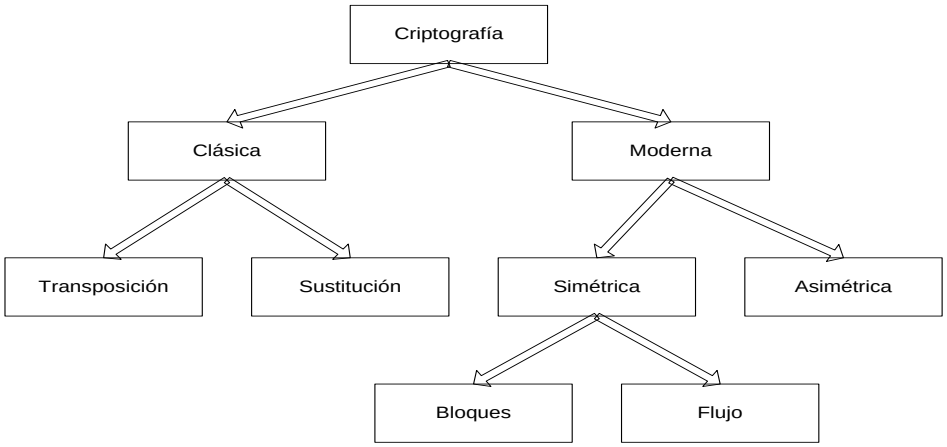
Figura 24. Disco cifrador de Alberti.



(ITESCAM, 2004)

La criptografía moderna se clasifica en clásica que se basa en modelos de transposición y substitución y moderna que se basa en claves simétricas y asimétricas. En la figura 25 se ilustra la clasificación que tiene la criptografía hoy en día.

Figura 25. Clasificación de la Criptografía.



(Granados Paredes, 2006)

4.4 ESTADO DEL ARTE

La criptografía es un arte muy antiguo que se remonta a civilizaciones como la griega o egipcia, que consiste en ocultar la información y que sea solo interpretada por los destinatarios originales. Hoy en día se ha convertido en una ciencia, debido a los procesos matemáticos-lógicos que este envuelve, su forma de aplicación ha cambiado desde procesos rudimentarios como el uso de figuras y objetos, el uso de máquinas mecánicas hasta el uso de los computadores que usan lenguajes binarios y en un futuro el uso de computadores cuánticos donde la transacción se de en bits cuánticos codificados en fotones.

En 1976 del artículo "New Directions in Cryptography" de Whitfield Diffie y Martin E. Hellman nace la idea de los sistemas criptográficos asimétricos, sistemas encargados de establecer claves simétricas, para evitar la distribución de claves mediante canales seguros (Diffie & Hellman, 1976).

En esta lectura Diffie and Hellman mencionan sobre un nuevo uso en claves o llaves para minimizar el uso de canales seguros de distribución de claves, por supuesto estos canales de distribución no son usados para la demás información, simplemente porque no pueden con la carga y representa un retardo en la información. También los nuevos avances ofrecen un equivalente de una firma. Diffie and Hellman ofrecen la clave o llave pública como solución de distribución de claves, el funcionamiento de esta consiste en que cada usuario tiene dos claves, una privada y una pública, la primera es una clave la cual solo el usuario tiene acceso y la segunda es la clave que está dispuesta al público. El procedimiento consiste: un usuario A quiere enviar un mensaje al usuario B, entonces A hace la encriptación del mensaje mediante la clave pública de B, una vez que B recibe el mensaje hace el descifrado mediante su clave privada, de esta manera la necesidad de canales seguros es suprimida (Diffie & Hellman, 1976).

La autenticidad es un problema pertinente en las comunicaciones, como en el ámbito financiero. Por ello debe existir un equivalente de firmas como se hace tradicionalmente, para asegurar la autenticad de los documentos, tal equivalente se le conoce como firma digital. La firma digital consiste: el usuario A quiere enviar un documento al usuario B y comprobarle a este que el mensaje es auténtico de A, entonces A hace la encriptación mediante su clave privada (y no la pública de B). Una vez que B recibe el documento, este realiza el descifrado mediante la clave pública de A, si el documento es claro para B, es porque sí proviene de A.

El en el año 1977 el cifrador DES fue desarrollado por IBM para ser utilizado por el gobierno de los Estados Unidos de América, se basa en técnicas de substitución y transposición, el cifrado y descifrado se hace mediante una llave o clave de 54 bits de longitud, a esto es importante mencionar que mayor sea el número de bits de la

clave, mayor es la seguridad, ya que para los criptoanalistas el proceso es más tedioso debido a los recursos que pueda requerir para acceder a la información (Mancilla Tello, 1995).

Ya siendo claro el diseño del cifrador DES, se procede a elaborar el algoritmo en pseudo-código. Para el algoritmo RSA, el diseño utiliza técnicas basadas en una función exponencial y una función módulo, el autor explica el diseño de este y lo implementa en pseudo-código. Los modelos implementados por el autor se acomodan a los requerimientos demandados hoy en día, la autenticidad y secretividad de los datos. Según el autor el cifrador DES es de mayor acogida en el mundo, como en Guatemala por el conocimiento que se tiene de este, por lo que a un nuevo criptosistema asimétrico (de mayor seguridad) le es más difícil su comercialización, por el poco conocimiento, lentitud y la falta de un protocolo estandarizado como el DES. Esto es válido para cuando el autor realizó la tesis, pero hoy en día sí se cuenta con un nuevo estándar para cifrado de datos, el AES (Advance Encryption Standard) como consecuencia de la intrusión en el DES. En esta tesis el autor define en forma clara y sin divagar de conceptos básicos de criptografía y métodos hasta el momento conocidos para el cifrado de datos, donde dicho producto se le denomina criptograma. Aunque prácticamente el autor se enfoca sobre la criptografía, hace mención del criptoanálisis, que el conjunto de las dos se le conoce como criptología. La criptografía es el arte de hacer la información ininteligible para los intrusos y el criptoanálisis es la encargada de interpretar estos mensajes sin algún tipo de autorización. Al tener en claro sobre el área que trabaja el autor, los métodos usados por este son técnicas conocidas que se han elaborado a través de la historia para la seguridad de la información. El autor quiere implementar algún tipo de cifrado de datos, o simplemente quiere elaborar criptosistemas, para la elaboración de los criptosistemas opta por los diseños de cifrador DES (Data Encryption Standard) y RSA (Rivest, Shamir y Adleman) (Mancilla Tello, 1995).

En el trabajo de Ibrahimy, Reaz, Asaduzzaman y Hussain titulado " FPGA Implementation of RSA Encryption Engine with Flexible Key Size " se resalta las características únicas que tiene la encriptación RSA al ser modelado sobre VHDL al soportar múltiples tamaños de claves que permite su implementación en sistemas que manejen distintos niveles de seguridad. La FPGA es una gran herramienta, pues en este trabajo, sus velocidades de trabajos son considerablemente altas, como cifrar con una clave de 1024 bits en una velocidad por debajo de los 41,585us. El trabajo enfoca el desarrollo del algoritmo RSA como una solución para una comunicación de voz segura. Lo interesante de este es que se busca desarrollar el algoritmo de tal forma que sea más eficiente que como normalmente se desarrollaría matemáticamente. A nivel de procesamiento es reducir las operaciones mediante sumas y restas. Lo importante de este trabajo es que el uso de la FPGA, por ser a nivel de hardware, la preferencia sobre este es la velocidad de procesamiento más que la seguridad, por lo que su uso en el mundo real es eficiente para desarrollar

problemas de mayor dificultad como lo es la encriptación RSA (Ibrahimy, Reaz, Asaduzzaman, & Hussain, 2007).

Durante la segunda guerra mundial se usaron máquinas mecánicas para el cifrado de información que se basaba en sustituciones y transposiciones, estas máquinas principalmente realizaba el cifrado con rotores, sin embargo se lograron romper los códigos de estas máquinas (Kruh & Deavours, 2002). El único código que no se logró romper durante esta época fue el utilizado por los estadounidenses que se basaba en el uso de códigos en navajo (una lengua amerindia), a las personas encargadas de realizar el cifrado y descifrado de los mensajes se les conoce como locutores de claves (National Security Agency, 2008).

En el documento "FPGA Implementation(s) of a Scalable Encryption Algorithm" los investigadores quieren observar el comportamiento y desempeño de este al implementarlo en hardware mediante el uso de una FPGA y comparando este junto con AES e ICEBERG. Los algoritmos implementados usando estructura loop, la propuesta por los investigadores presenta flexibilidad para cualquier parámetro de SEA y muestra ventajas significantes al usar VHDL genérico, y que debido a su característica de ser escalable, entonces el tamaño de los parámetros como texto, clave y bus de datos no cambian el código del lenguaje, puede ser puede ser re-implementado para dicho propósito haciendo uso de síntesis y herramientas de implementación. El desempeño en hardware es el menor ante los algoritmos que se comparan, debido al bajo consumo en área (LUTs), y gran parte de ello se debe a que SEA trabaja en paralelo tanto para las rondas de clave y rondas de cifrado/ descifrado y que su diseño fue dirigido a rutinas que impliquen operaciones como XOR, AND, rotación de palabra y adición modular (Macé, Standaert, & Quisquater, 2008).

SEA, por su bajo costo en hardware, tiene cabida para implementaciones de bajo consumo de poder en ASICs.

En el trabajo de Xianwei, Erhong, Li y Lang titulada "LUT-based FPGA Implementation of SMS4 / AES / Camellia" los autores mediante la prueba de tres distintos algoritmos de cifrado, miden el desempeño de los dispositivos FPGA que difieren por sus características. El desempeño está basado en dos parámetros principales como el mismo rendimiento del algoritmo en la implementación y el área que abarca en el dispositivo. Para saber la influencia sobre el área, el estudio se hace con dos distintos tamaños de LUT (Look Up Table) correspondientes a los dispositivos FPGA, que son unos índices usados en computación para la reducción de procesamientos, pues hay una respuesta según la función de entrada. Se hace una estructura de retroalimentación para el diseño, que de tal forma puede conseguir un balance entre velocidad y área y también así conseguir una comparación más precisa entre los algoritmos. Para la descripción del circuito se

hace uso de RTL-VHDL, que en principio sería ideal usar librerías por eficiencia, pero estas están sujetas a los dispositivos, por lo que es idóneo desarrollarlo en un lenguaje que aplique para todos los dispositivos como el seleccionado VHDL'93. Las FPGAs usadas fueron Ciclone II y Stratix II para las pruebas, y se concluye que la Stratix II es la que más se ajusta a las funciones de los cifradores, esto se debe a que tiene mayor ancho de LUT en los ALM (Adaptive Logic Module), encargados de hacer uso eficiente de los recursos lógicos, y además que presenta compatibilidad con arquitecturas de menor número de entradas, por ejemplo con la LUT de 4 entradas, que es la que dispone Ciclone II. En cuanto a los algoritmos, SMS4 tiene el menor costo en cuanto a hardware haciéndolo el idóneo para WLAN, y los demás que manejen Feistel presentan mayor eficiencia en hardware ya que por la similitud en los procesos puede compartir un mismo módulo para encriptación y descifrado (Xianwei, Erhong, Li, & Lang, 2008).

Hiremath & Suma en su trabajo titulado "Advance Encryption Standard Implemented on FPGA" explican que el AES (estándar de encriptación avanzada) es un cifrado que usa bloques de 128 bits y llaves de 128 bits, 196 bits y 256 bits, el surgimiento de este estándar se debe a que su antecesor el DES (Estándar de encriptación de datos), es débil por su llave de 56 bits y bloques de 64 bits y se volvía lento al implementarse sobre software. El AES usa el algoritmo Rijndael, escogido en el 2000 por su rapidez que presenta tanto en hardware y software, mejor rendimiento, seguridad y habilidad de implementación y flexibilidad. El algoritmo Rijndael hace un número de rondas para cifrado y descifrado, donde hace operaciones sobre matrices de 4x4. Se hace la implementación del AES sobre una FPGA Xilinx Spartan 3, por lo tanto es a nivel de hardware. El Diseño se basa en dos bloques principales: El Controlador Principal que se enfoca a la interfaz con el usuario y el núcleo AES, estos trabajan síncronamente. La implementación final demanda un 70% de los recursos de la FPGA con un total 75404kB usados en memoria. La síntesis de este arroja porcentajes acerca de la utilización del dispositivo, que de tal forma se observa la demanda de recursos para el empleo del algoritmo (Hiremath & Suma, 2009).

En el documento " A Hardware-Oriented Fast Encryption Keystream Generator for Digital Rights Management" los autores argumentan que actualmente no se cuenta con algún tipo de cifrado estandarizado sobre DRM (Manejo de Derechos Digitales), y aquellos que se utilizan son débiles debido a su tamaño en cifrado. Para entender mejor el tema DRM se refiere a tecnologías de control de acceso usado como por ejemplo por fabricantes de hardware para cifrar sobre contenidos. Tradicionalmente los generadores de clave de flujo se han desarrollado sobre registros de desplazamiento, el más típico de ellos usa LFSR (Registros de desplazamiento de retroalimentación lineal), pero hoy en día son inseguros debido a la capacidad computacional disponible. FCSR (Registros de desplazamiento de retroalimentación con acarreo) son una solución para remplazar LFSR que son débiles por su estructura matemática, además FCSR no presenta linealidad. Los

autores presentan un modelo que presenta aleatoriedad perfecta mediante del uso de un FCSR diseñado para garantizar un nivel de seguridad alto y el uso de tres filtros lineales. Los resultados se ven reflejados mediante el uso de una herramienta de NIST que determina la aleatoriedad del generador, que al ser comparados con otros modelos existentes, es competente (Ma Jianfeng & Dong Lihua, 2009).

En la lectura "Rivest unlocks cryptography's past, looks toward future" el profesor Rivest habla de la necesidad de implementar a un futuro nuevos sistemas criptográficos, debido a que los existentes implementados pueden ser interpretados, lo que los harían obsoletos, el problema para los criptoanalistas es el encontrar el método idóneo para romper el código en el tiempo requerido, y tal método es esencial por las limitaciones de los recursos, como el tener computadores cuánticos u otro tipo procesamiento que se ve reflejado en su alto precio. La criptografía ha evolucionado de forma impresionante, que se remonta desde las civilizaciones de Egipto y Grecia con la excítala, como uno de los primeros artes de ocultar la información, donde el mensaje escrito poseía símbolos innecesarios que desaparecían al enrollarse en un rodillo de longitud y diámetro específico. Tal ha sido el avance de la criptografía que transcendió de un arte a una ciencia, una ciencia que envuelve matemáticas, estadística, ciencia computacional e ingeniería electrónica. La llegada del computador ha cambiado la forma de hacer criptografía, los criptosistemas se realizan lo más complejo posible, de esta forma se garantiza que interpretar estos tardaría muchos años, pero la evolución de la tecnología es inminente, como por ejemplo el sistema de encriptación RSA que es implementado en internet como solución en transacciones financieras y comunicaciones seguras, una forma de romperlo es mediante el hallazgo de factores primos de un número, se había dicho que los de un número de 129 bits tardaría millones de años en resolverse, 17 años después se resolvió en 8 meses con un equipo de 600 voluntarios, aun así es demasiado tiempo con respecto a la vida que tiene la información, pero no hay duda que los computadores cada vez son más capaces. Sin embargo el acceder a la información del criptograma, no garantiza la información original pues hay demás técnicas que son aplicadas para autenticidad como certificados digitales que usa firmas digitales, por lo tanto el uso de más técnicas mayor la seguridad de la información (Chandler, 2011).

En el trabajo de Cachin titulado "Storage encryption & key management" en el 2012, la envoltura de clave consiste en algoritmos de encriptación simétricos para encapsular claves de cifrado, en pocas palabras es cifrar claves de cifrado. Estos algoritmos son aplicados en redes de comunicación inseguras cuando se quiere transmitir claves o también proteger las claves en almacenamientos no fiables. El manejo de claves consiste en el manejo de claves de cifrado en un criptosistema, tareas como generación, almacenamiento, intercambio y remplazo de claves. Un bloque de cifrado ajustable es aquel donde hay una segunda entrada llamada “tweak” a la misma vez con la entrada de texto en claro o cifrado, lo que hace el

tweak junto con la clave es seleccionar la permutación calculada por el cifrado (Cachin, 2012).

En Latinoamérica, así como a nivel mundial se opta por la implementación de algoritmos populares como DES, AES, RSA, los cuales han tenido gran acogida, sin embargo DES ha sido sustituido por AES, o por lo menos como su reemplazo formal.

Hoy en día se cuenta con tecnologías como dispositivos y lenguajes de programación que son aplicados a nivel de hardware como VHDL.

Definitivamente existe una gran variedad de algoritmos de cifrado disponibles hoy en día, unos diseñados principalmente para software, otros para hardware o para ambos, buscando eficiencia en tiempo y procesamiento. Un claro ejemplo de esto fue el proyecto eSTREAM que de un conjunto de candidatos, se evaluó seguridad, y eficiencia en entornos de implementación. (ECRYPT, 2013)

En los últimos años las investigaciones se enfocan en hacer un manejo eficiente de los recursos de las tecnologías existentes, como las FPGAs, que al tratarse de hardware las velocidades de trabajo son altas, y el hacer una implementación adecuada, se puede extrapolar su función mediante otras herramientas sin afectar el código del lenguaje que estas usan (VHDL genérico).

Desarrollar algoritmos de cifrado es una opción, pero requiere un profundo estudio sobre las ramas de las matemáticas y sobre todo experiencia en criptografía y criptoanálisis, si se quiere que dicho algoritmo sea competente ante los actuales y que sea la solución ideal ante el problema propuesto.

Existe una gran variedad de algoritmos de cifrado disponibles hoy en día, unos diseñados para software, otros para hardware o para ambos, buscando eficiencia en tiempo y procesamiento.

El avance de la tecnología es otro factor que sobre la criptografía traería nuevos posibles desarrollos, como hacer el uso de fotones (nanotecnología) para cifrado de datos.

5. LIMITACIONES Y ALCANCES

5.1 LIMITACIONES

Como principal objetivo de hacer un análisis comparativo de cifradores en una FPGA de referencia Altera EP2C20F484C7, se especifica por la disponibilidad y único recurso de este tipo de dispositivos en la universidad, por lo que no se hará implementaciones sobre otro tipo de dispositivos programables.

Los cifradores a implementar están sujetos a características del modelo de la FPGA como unidades lógicas, velocidad operación, y capacidad de memoria.

El uso de la FPGA sirve de prototipo de un dispositivo final (circuito integrado) para la seguridad de datos en capa física, pero no deja de ser un prototipo, es una aproximación, más no un equivalente.

5.2 ALCANCES

VHDL es un lenguaje de programación diseñado para describir circuitos lógicos, es decir que hace posible la manufactura del circuito integrado a partir de un código, pero no es parte del proyecto la manufactura de dicho circuito integrado.

No se pretende implementar distintas versiones de los algoritmos de cifrado A5, W7 y Hélix, solo una versión será implementada. Con respecto a las claves no se generaran aleatoriamente, serán definidas por el desarrollador, para su respectiva comparación.

La implementación obedece al algoritmo de cifrado, pero no obedece a estándares ni recomendaciones de cómo implementar.

6. DISEÑO METODOLÓGICO

Se realizó una recopilación de información acerca de los sistemas criptográficos, que tipos de algoritmos se usan y como estos trabajan. En segundo lugar se buscó todo la información correspondiente al lenguaje de programación VHDL, sobre el cual se implementó el proyecto. A continuación se enuncian la serie de pasos realizados para el desarrollo del proyecto:

- Se recopiló y describió teoría y conceptos de criptología, sobre todo en criptografía, dado que los cifradores corresponden a esta área.

- Se recopiló información existente sobre VHDL, lenguaje de programación en el cual se implementaron los sistemas cifradores.

- Se seleccionó tres algoritmos de cifrado de flujo A5/1, W7 y Hélix. A5/1 se seleccionó por ser un cifrador con una aplicación específica en las telecomunicaciones, al igual que en medio de la investigación se encontró que W7 es una propuesta en reemplazo al algoritmo A5/1 por cuestiones de seguridad. Y Hélix por ser un cifrador rápido y eficiente tanto en hardware como software.

- Se seleccionaron tres variables de comparación posible mediante el software Quartus II Web Edition, los cuales son: Elementos lógicos utilizados, potencia térmica disipada y velocidad de desempeño.

- Se diseñaron los sistemas cifradores en VHDL, definiendo la lógica secuencial mediante máquinas de estado.

- Se simuló mediante vectores de prueba los sistemas desarrollados.

- Se sintetizó los sistemas desarrollados para la FPGA EP2C20F484C7N.

- Se realizó el análisis comparativo sobre sobre la simulación y resultados de compilación del software Quartus II Web Edition.

7. DESARROLLO

Esta sección inicia con la identificación de las características, ventajas y desventajas de los algoritmos de cifrado descritos en el marco referencial.

Posteriormente se identifica los requerimientos para diseñar cada uno de los sistemas cifradores en VHDL. Se identifican los bloques funcionales de cada uno de los cifradores, para estos bloques se diseñan las entidades que al final se ensamblan para formar el cifrador. Se hace de este modo por eficiencia, para aprovechar mejor los recursos y lograr mejores velocidades de desempeño. En la identificación de bloques se tuvo en cuenta que debería funcionar de forma concurrente o secuencial.

Los bloques identificados son descritos en VHDL, y sintetizados para la FPGA, es decir que se le especifica al compilador de Quartus II la referencia de la FPGA para dar cumplimiento a la secuencia de diseño de VHDL. (Ver la figura 21)

En pruebas y resultados se comprueba el correcto funcionamiento de los cifradores utilizando el simulador de Quartus II y finalmente se prueba el correcto funcionamiento sobre la FPGA.

7.1 VENTAJAS Y DESVENTAJAS DEL CIFRADOR A5/1

Este algoritmo se describe en el marco teórico de este documento, a partir de esta descripción y de lo que se encontró a lo largo de esta investigación, en la siguiente tabla se resume las características, ventajas y desventajas del algoritmo de cifrado A5/1.

Tabla 4. Características, ventajas y desventajas de cifrador A5/1.

ALGORITMO A5 / 1		
CARACTERISTICAS	VENTAJAS	DESVENTAJAS
• Algoritmo de cifrado simétrico. • Para cifrar 228 bits de información genera una secuencia de cifrado a partir de	• Basado en LFSRs, que tienen propiedades seudo-aleatorias. • Algoritmo idóneo para hardware, usa registros relativamente pequeños y	• Poca documentación acerca del algoritmo. Además poco fiable. La ETSI no tiene un

una clave de 86 bits. Esta clave está constituida por una clave privada de 64 bits y el número de trama de 22 bits • Para cifrar 228 bits consume 415 ciclos de máquina, lo que se resume que por bit consume 1.82 ciclos de máquina. • Usado en GSM para el cifrado de voz.	operaciones lógicas booleanas. • El algoritmo no ha de consumir tantos recursos si se define su lógica secuencial por una máquina de estados. • No hay necesidad de almacenar toda la trama para el cifrado, se puede hacer uso de buffers pequeños. • La secuencia de cifrado no es dependiente del texto plano, lo que si en una transmisión un dato se pierde, no afecta al resto que se transmite.	documento formal gratuito. • Al final de cada trama se debe cargar nuevamente la clave de 86 bits, ya que 22 bits de esta representan el número de trama. • Para el poder computacional disponible hoy en día 86 bits de clave, es poco, se recomienda una clave de mayor valor.

7.2 VENTAJAS Y DESVENTAJAS DEL CIFRADOR W7

De manera similar, la siguiente tabla resume las características, ventajas y desventajas del algoritmo de cifrado W7, tomando como base la descripción realizada en el marco teórico.

Tabla 5. Características, ventajas y desventajas de cifrador W7.

ALGORITMO W7		
CARACTERISTICAS	VENTAJAS	DESVENTAJAS
• Algoritmo de cifrado simétrico	• Basado en LFSRs, que tienen propiedades seudo-aleatorias. • Análogo al A5/1, es más seguro al utilizar una	• Al operar con varias celdas en paralelo, requiere más recursos.

• Realiza el cifrado por secuencias de byte. • Consume 1032 ciclos de máquina para su iniciación, la cual basa en una clave simétrica de 128 bits. • Carga la clave simétrica paralelamente.	función de filtrado en su salida. • De fácil implementación. Basado en celdas, cada celda está compuesta de registros pequeños y operaciones lógicas booleanas. • Definiendo hardware por máquina de estados, el algoritmo no ha de consumir tantos recursos. • La secuencia de cifrado no es dependiente del texto plano, lo que si en una transmisión un dato se pierde, no afecta al resto que se transmite. • Más rápido al cifrar por bytes. • No está restringido a un número de bits específico, o tamaño de trama.	• De la IETF, fue abandonado en el 2003, lo que sugiere que no es muy competente, por lo menos con los del portafolio eSTREAM.

7.3 VENTAJAS Y DESVENTAJAS DEL CIFRADOR HÉLIX

Al igual que los anteriores algoritmos de cifrado, en la siguiente tabla se resume las características, ventajas y desventajas del algoritmo de cifrado Hélix.

Tabla 6. Características, ventajas y desventajas de cifrador Hélix.

ALGORITMO HÉLIX		
CARACTERISTICAS	VENTAJAS	DESVENTAJAS
• Cifra palabras de 32 bits. • El cifrado es dependiente del texto plano. • La iniciación del cifrador requiere de 15 ciclos de máquina. • Ofrece un mensaje de autenticación a cambio de 12 ciclos de máquina demás.	• Opera con longitudes de texto plano de 0 hasta 2^64 bytes. • Permite el uso de una clave variable en longitud de 0 hasta 256 bits. • Ofrece mensaje autenticación	• No es cifrador de flujo síncrono, por lo que la pérdida de un dato altera un conjunto de datos

7.4 DISEÑO DE ENTIDADES PARA EL CIFRADOR A5/1

Para el cifrador A5/1 se definen las siguientes entidades para su construcción:

FUNCION MAYORITARIA: Esta entidad, a partir de los habilitadores de reloj de cada LFSR, evalúa si le concede su desplazamiento y solo ocurrirá si el valor de su bit habilitador coindice con valor que es mayoría, sea 1 o 0.

La entidad cuenta con tres entradas y una salida, sus funciones son:

Entrada “enai”: Habilita la función de bloque. Si está activo el bloque evalúa los bits provenientes de los LFSRs, los cuales tiene como entradas para determinar si habilita el reloj para estos. Si esta desactivo todos los LFSR reciben el reloj del sistema.

Entrada “iclk”: Esta estrada es el reloj de sistema, que se utiliza para dar reloj a los LFSRs.

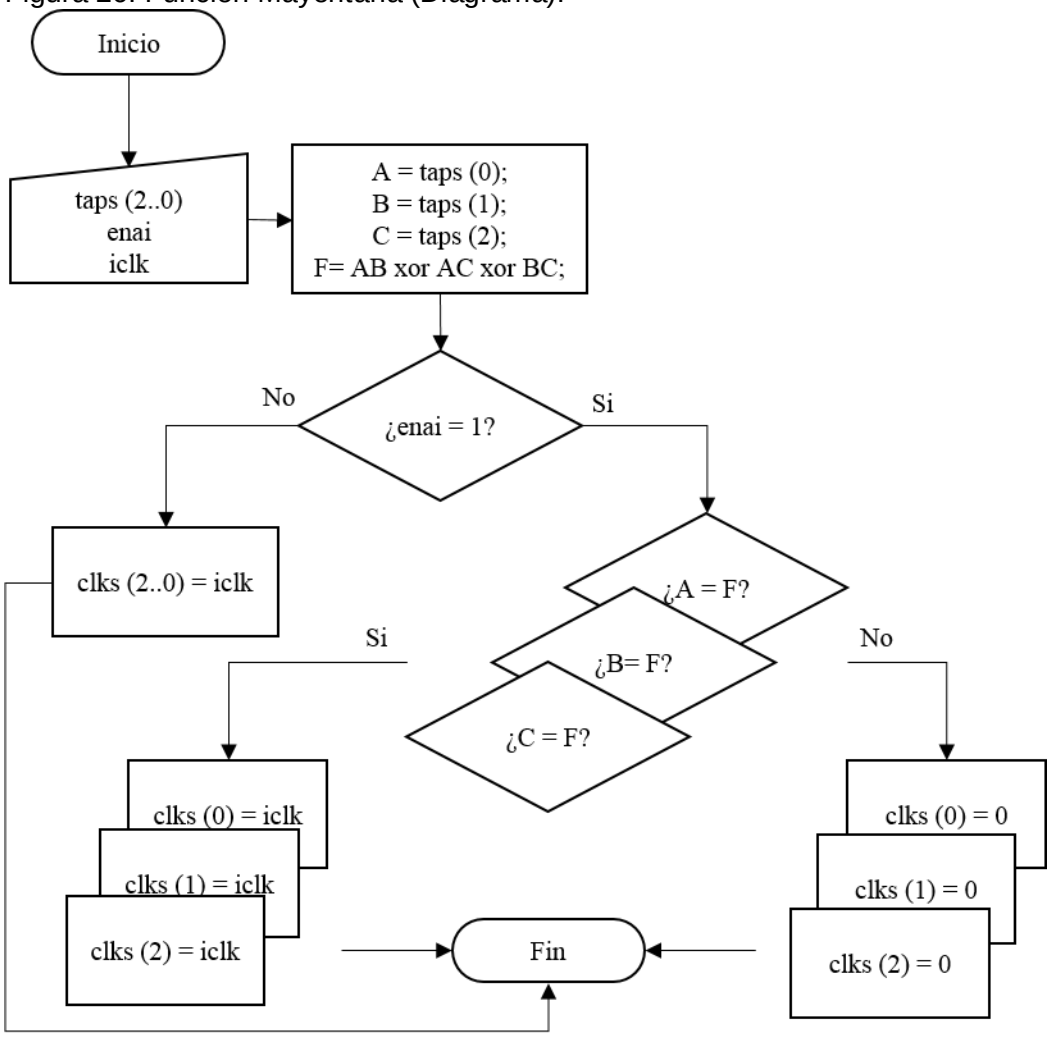
Entrada “taps”: Este recibe el conjunto de bits selectos de los todos los LFSRs del sistema, de este conjunto se determina a quienes se les habilita el reloj o no.

Salida “clks”: Envia señal de reloj a los LFSRs

El código VHDL es netamente combinacional, las operaciones en la arquitectura fueron determinadas mediante mapas de Karnaugh con SOP (suma de productos). (Ver anexos).

En la figura 26 un diagrama de flujo ilustra la funcionalidad de la entidad.

Figura 26. Función Mayoritaria (Diagrama).



7.4.1 LFSR genérico

El tipo de dato generic en VHDL, es útil para hacer cambios en el hardware descrito, sin la necesidad de hacer una gran cantidad de cambios en el código. Pues el uso debido de este tipo de datos evita que se altere la funcionalidad del código y poder hacer de este, un código reutilizable.

La entidad LFSR descrito en VHDL opera cada vez que detecta una señal de reloj, tiene 4 entradas y dos salidas. Las funcionalidades de cada puerto son las siguientes:

Entrada “clk”: Habilita la operación del bloque LFSR, si el bloque no detecta eventos, simplemente mantiene su estado.

Entrada “rst”: Borra el registro, es decir pone ceros a lo largo del registro.

Entrada “inp”: Es el puerto por donde accede los datos.

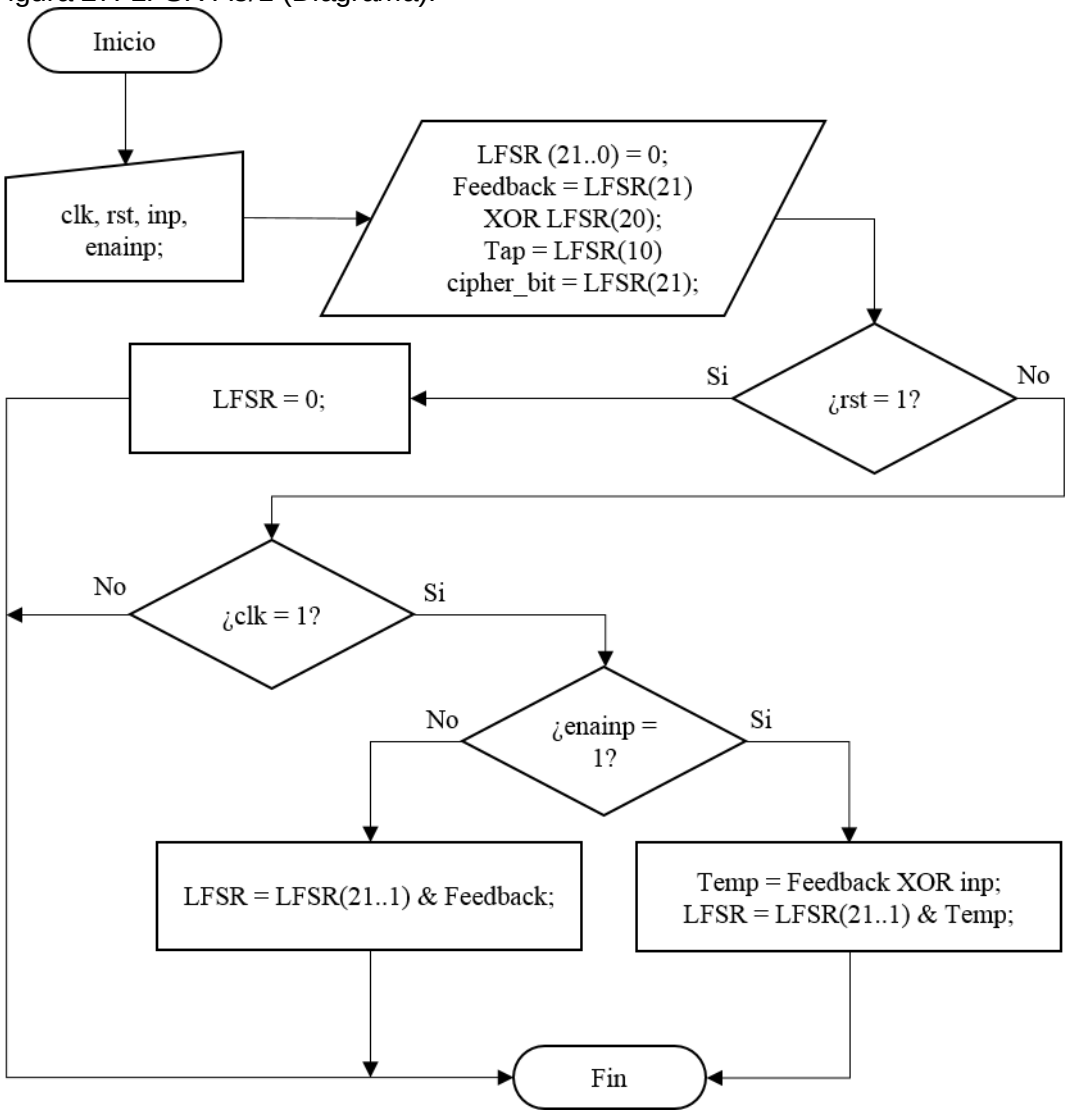
Entrada “enainp”: Habilita la entrada de datos al LFSR, que internamente se opera (XOR) junto con la función de retroalimentación del LFSR, guarda el resultado en el LSB y desplaza el registro (SLL). Si está desactivado se escribe el resultado de la función de retroalimentación en el LSB y desplaza el registro (SLL).

Salida “tapbit”: Bit necesario para definir la operación del bloque. La evaluación es realizada por el bloque “función mayoritaria”.

Salida “cipher_bit”: Es el MSB del LFSR, necesario para la operación de algoritmo.

En la figura 27 un diagrama de flujo ilustra la funcionalidad de la entidad.

Figura 27. LFSR A5/1 (Diagrama).



7.4.2 Celda

Es la unión de las entidades LFSR y Función Mayoritaria. La razón de ser de la unión de estas entidades es por la dependencia de uno con el otro, y las entradas es común que tienen. A esta unión se le añade una operación XOR entre las salidas de cada LFSR (Aquellas denominadas "cipher_bit"). La celda está constituida de 4 entradas y una salida:

Entrada "cell_clk": Es el reloj de los LFSR y de la función mayoritaria.

Entrada "cell_rst": Reinicia los LFSR, toman el valor de cero lógico.

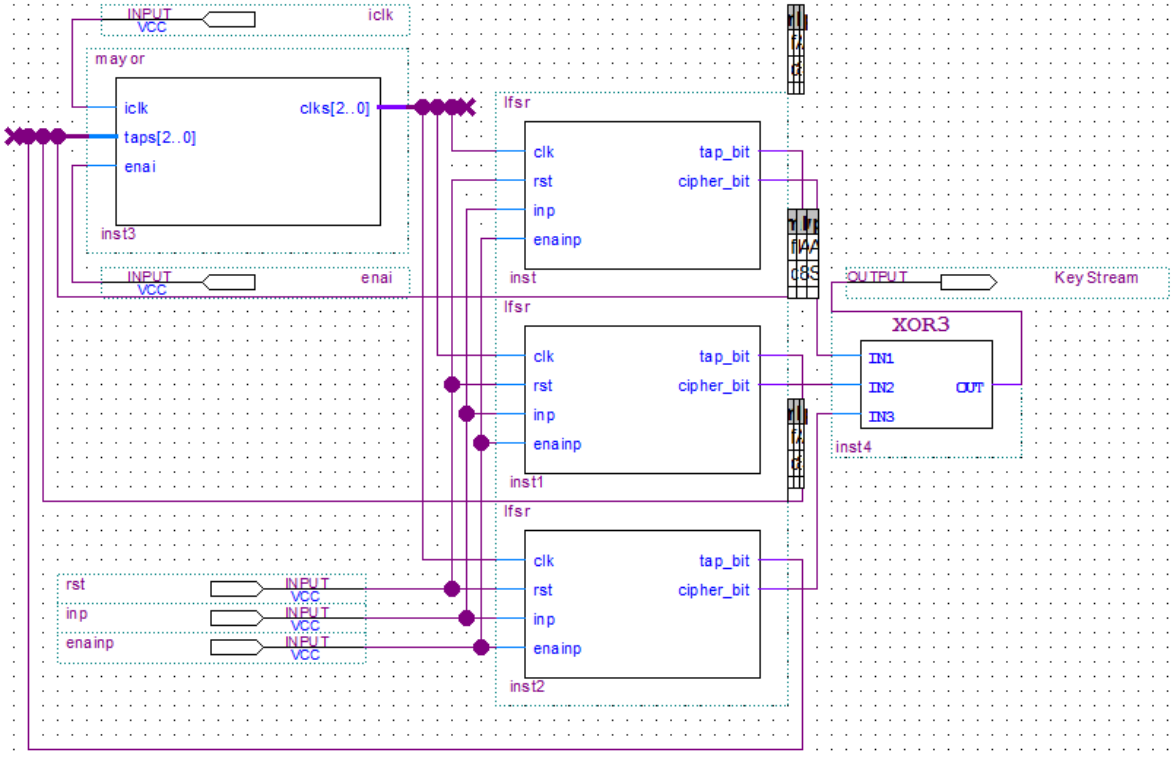
Entrada "cell_data": Por este ingresa los datos de la clave y el número de trama.

Entrada "cell_data_ena": Indica si deja el paso de datos a los LFSRs, al igual que indica a la función mayoritaria si debe entregar reloj regular o irregular a los LFSRs. (El reloj regular solo se debe entregar cuando se deja ingresar datos).

Salida "ks_bit": Esta salida es la operación OR exclusiva de las salidas de los tres LFSR, este resultado es el bit utilizado para el cifrado.

En la figura 28 se ilustra la interconexión entre las distintas entidades para la construcción de la celda.

Figura 28. Construcción de la celda.



7.4.3 Memoria de clave y trama

La función del bloque es transmitir los bits de la clave y el número de trama. Este está constituido de dos entradas y una salida, y sus funciones son:

Entrada “clk”: Es el reloj del sistema, un ciclo de esta señal equivale a un bit de salida.

Entrada “rst”: Reinicia la entidad.

Salida “d”: Es la salida de datos.

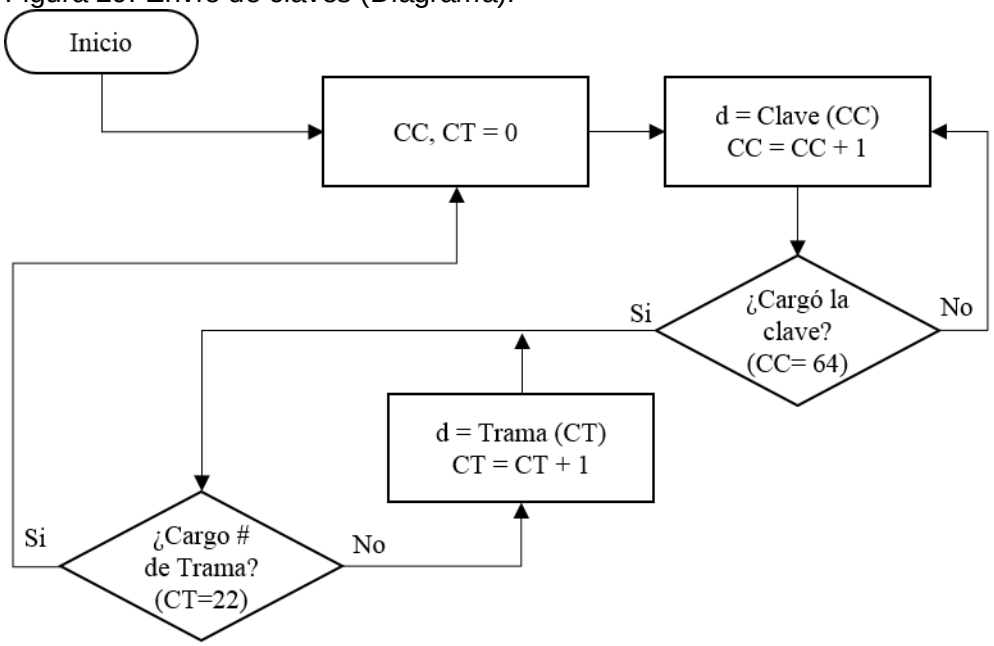
Internamente la entidad cuenta con dos memorias que almacena la clave y el número de trama.

Registro "key": Contiene la clave de 64 bits.

Registro "iniframe": Contiene el número de trama de 22 bits.

La entidad se desarrolla para enviar de forma serial los datos de la clave y el número de trama, los cuales se leen de los registros memoria, el diagrama de flujo de la figura 29 ilustra su funcionalidad.

Figura 29. Envío de claves (Diagrama).



7.4.4 Memoria de secuencia de cifrado

Esta entidad es la encargada de almacenar la secuencia de bits entregados por los LFSR para cifrar los 228 bits del texto plano. Está constituida de tres entradas y una salida, y sus funciones son:

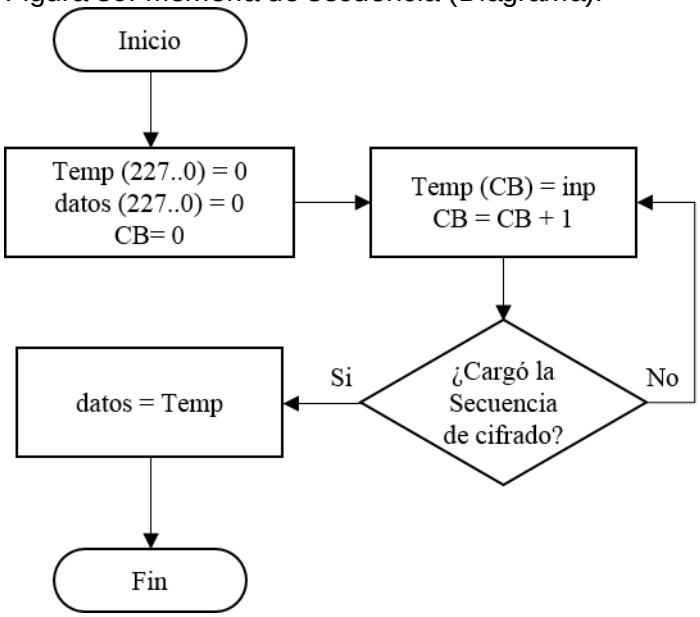
Entrada "inp": Es por donde ingresa los la secuencia de bits para el cifrado.

Entrada "rst": Reinicia la entidad, dejando su memoria en cero.

Entrada "clk": Es el reloj del sistema, un ciclo de esta señal equivale a guardar un bit en la memoria.

Salida "datos": Entrega la secuencia de cifrado una vez haya concluido 228 ciclos de reloj.

Figura 30. Memoria de secuencia (Diagrama).



7.4.5 Unidad de control

Esta entidad se basa en una máquina de estados que determina los estados de las distintas entidades. Es el responsable de dar cumplimiento al algoritmo. La unidad de control está constituida de tres entradas y cuatro salidas, y sus funciones son:

Entrada "rst": Reinicia la unidad de control.

Entrada "clk": Es el reloj del sistema, este mismo reloj es usado por la entidad de control para brindar reloj a las demás, para que estos funcionen.

Entrada "go"; Definido por el desarrollador para pruebas, si permanece en 1, el cifrador funcionará normalmente, de no serlo, la máquina de estados se detiene en el último estado, que es cuando se halla generado toda la secuencia de cifrado.

Salida "cell_clk": Habilita el reloj para la celda.

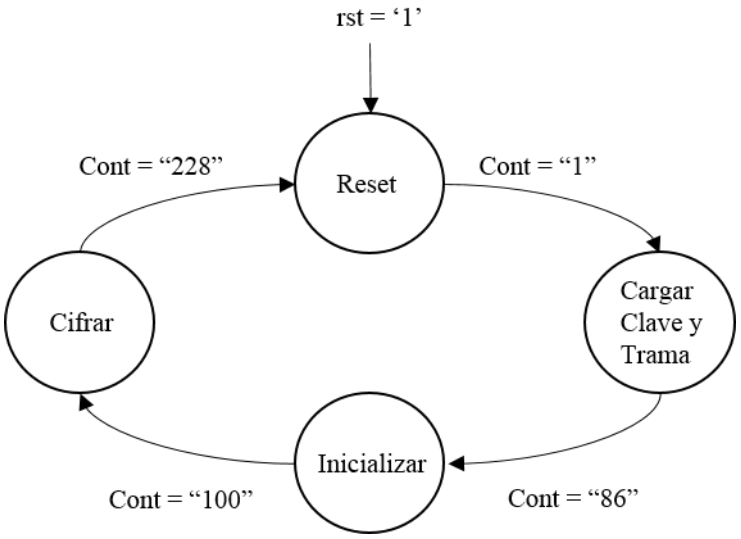
Salida "cell_rst": Reinicia los LFSR, toman el valor de cero lógico. Al igual que reinicia la memoria de secuencia de cifrado y la memoria de clave y trama.

Salida "cell_data_ena": Indica si deja el paso de datos a los LFSRs, al igual que indica a la función mayoritaria si debe entregar reloj regular o irregular a los LFSRs. (El reloj regular solo se debe entregar cuando se deja ingresar datos).

Salida "buf_ena": Habilita la escritura en la memoria de secuencia de cifrado, al enviar la señal de reloj.

La máquina de estados describe el paso a paso para poder cifrar los datos, la cual se ilustra en la figura 31.

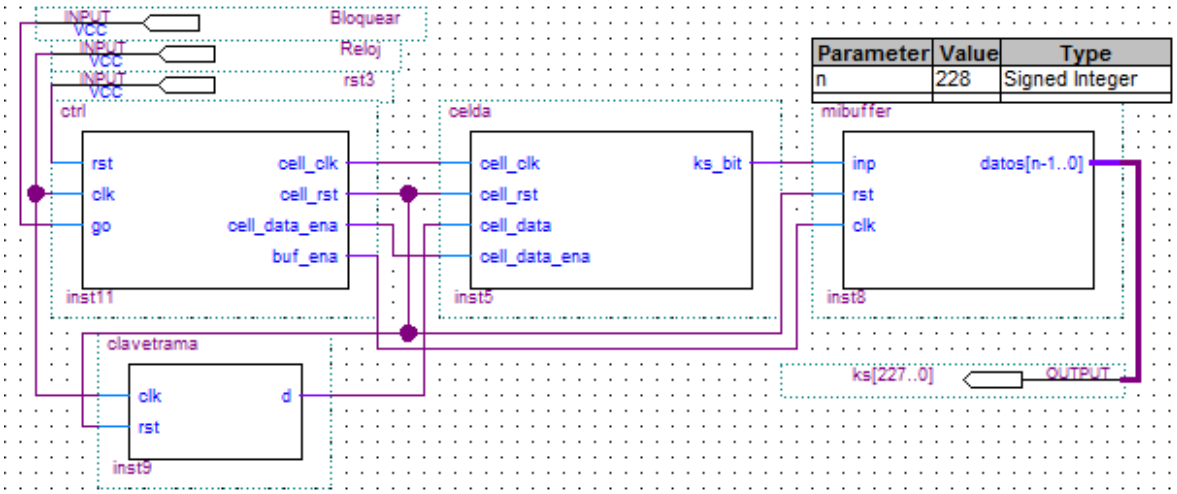
Figura 31. Máquina de estados (Diagrama).



7.4.6 Interconexión

En la figura 32 se ilustra la conexión entre las entidades diseñadas.

Figura 32. Interconexión entre entidades de A5/1.



7.5 DISEÑO DE ENTIDADES PARA EL CIFRADOR W7

7.5.1 Función Mayoritaria

Esta entidad es idéntica a la descrita para el algoritmo A5/1, a excepción de que su habilitador siempre está activo.

7.5.2 LFSR Genérico

La entidad LFSR descrito en VHDL opera cada vez que detecta una señal de reloj, tiene 3 entradas y dos salidas, la funcionalidad de cada puerto son las siguientes:

Entrada “clk”: Habilita la operación del bloque LFSR, si el bloque no detecta eventos, simplemente mantiene su estado.

Entrada “rst”: Carga el registro con el valor de la clave.

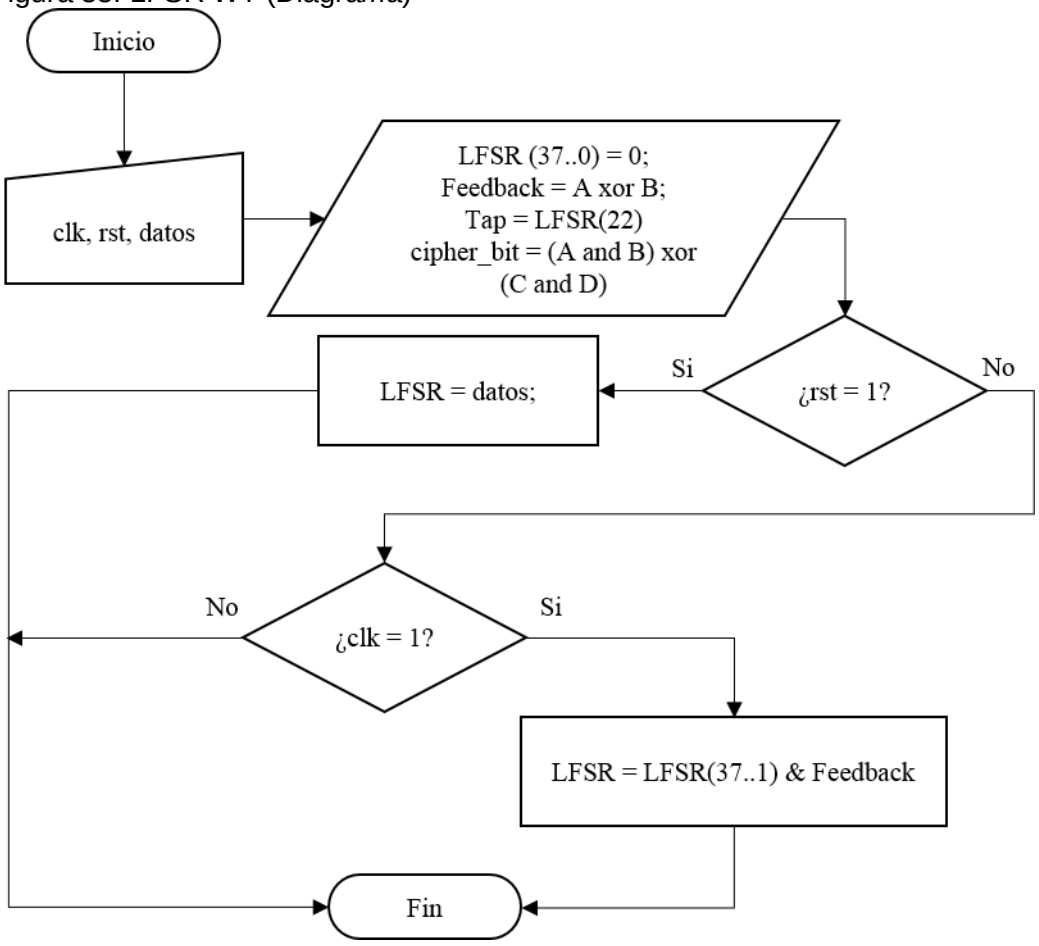
Entrada “datos”: Es el puerto por donde accede los datos, tiene un ancho de 128 bits.

Salida “tap_bit”: Bit necesario para definir la operación del bloque. La evaluación es realizada por el bloque “función mayoritaria”.

Salida “cipher_bit”: Es el MSB del LFSR, necesario para la operación de algoritmo.

En la figura 33 un diagrama de flujo muestra la funcionalidad del LFSR.

Figura 33. LFSR W7 (Diagrama)



7.5.3 Celda

Es la unión de las entidades LFSR y Función Mayoritaria. Esta constituido de 4 entradas y una salida:

Entrada "cell_clk": Es el reloj de los LFSR y de la función mayoritaria.

Entrada "cell_rst": Reinicia los LFSR, toman el valor de cero lógico.

Entrada "cell_datos": Por este ingresa los datos de la clave y el número de trama.

Salida "ks_bit": Esta salida en la operación OR exclusiva de las salidas de los tres LFSR, es decir el bit de cifrado.

Así como se ilustro la construcción de la celda para A5/1 en la figura 28, para W7 la interconexión entre las distintas entidades es exactamente igual.

7.5.4 Unidad de control

Esta entidad se basa en una máquina de estados que determina los estados de las distintas entidades, es el responsable de dar cumplimiento al algoritmo. Está constituida de tres entradas y cuatro salidas, y sus funciones son:

Entrada "main_rst": Reinicia la unidad de control.

Entrada "main_clk": Es el reloj del sistema, este mismo reloj es usado por la entidad de control para brindar reloj a las demás para que estos funcionen.

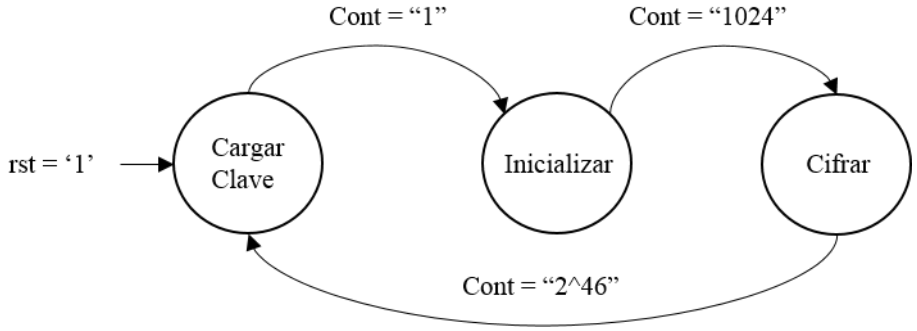
Entrada "go"; Definido por el desarrollador para pruebas, si permanece en 1, el cifrador funcionará normalmente, de no serlo, la máquina de estados se detiene en el último estado, que es cuando se halla generado toda la secuencia de cifrado.

Salida "cell_clk": Habilita el reloj para la celda.

Salida "cell_rst": Reinicia los LFSR, toman el valor de cero lógico. Al igual que reinicia la memoria de secuencia de cifrado y la memoria de clave y trama.

Salida "buf_ena": Habilita la escritura en la memoria de secuencia de cifrado, al enviar la señal de reloj.

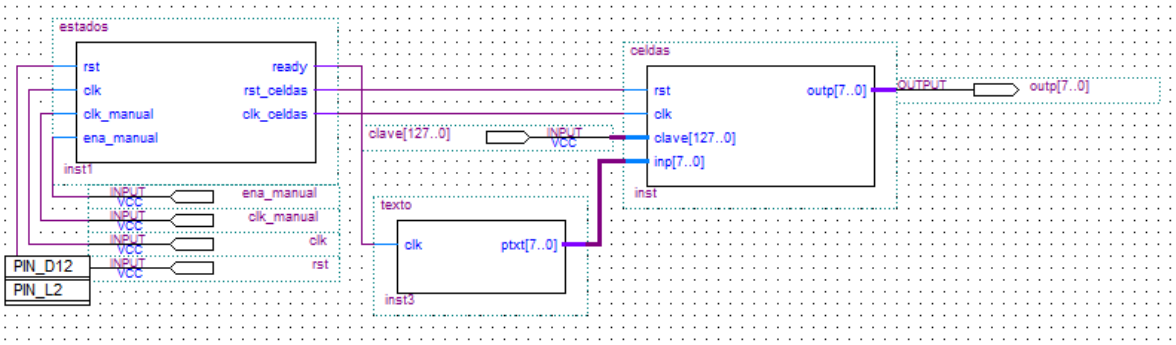
Figura 34. Máquina de estados W7 (Diagrama).



7.5.5 Interconexión

En la figura 35 se ilustra la conexión entre las entidades diseñadas.

Figura 35. Interconexión entre entidades de W7.



7.6 DISEÑO DE ENTIDADES PARA EL CIFRADOR HÉLIX

El cifrador Hélix se define en una sola entidad en VHDL y se diseña las siguientes funciones:

Rol: Función que hace rotación lógica hacia la izquierda un número determinado de veces.

To dword: Convierte un vector binario en un vector de palabras de 32 bits. Se diseña para hacer más simple el manejo del código.

To std logic vector: convierte un entero en un vector binario. Se diseña debido a la incompatibilidad entre los tipos de datos predefinidos en VHDL.

Xor: Se diseña nuevamente para permitir esta operación entre los tipos de datos definidos por el usuario.

Mitad Hélix: Es la mitad del bloque Hélix y se diseña de esta forma por eficiencia, ya que su otra mitad es idéntica. Cuando se habla del bloque Hélix, se habla del uso doble de esta función.

F: Esta función define los parámetros de entrada hacia el bloque Hélix, requerido por la función "wkey" para la generación de la clave de trabajo.

Wkey: Define la secuencia lógica para generar la clave de trabajo, tomando como parámetros la clave y su longitud en bytes.

Xnonce: Recibe el vector binario que representa el nonce y devuelve el nonce extendido con 128 bits demás.

Xicero: Recibe la clave privada y el número de ronda para seleccionar y devolver una de las 8 palabras de 32 bits de la clave de trabajo.

Xiuno: Recibe la clave de trabajo y el nonce extendido, y devuelve una palabra de 32 bits la cual se genera mediante una operación no lineal entre la clave de trabajo y el nonce extendido.

Zini: Establece los cinco estados para el bloque Hélix para la iniciación del cifrador.

A partir de estas funciones se crea una máquina de estados, responsable de inicializar el cifrador y permitir el cifrado de datos el cual ingresa en palabras de 32 bits.

La máquina de estados consta de 2 estados. El primero tiene una duración de 15 ciclos de reloj para inicializar el cifrador, y el segundo estado habilita el paso de datos hacia el cifrador en palabras de 32 bits por ciclo de reloj. La máquina de estados consta con los siguientes puertos para su funcionamiento:

Entrada "clk": Es el reloj del sistema, este mismo reloj es usado por la máquina de estados para brindar reloj a la memoria de que contiene el texto plano, esta memoria envía el texto plano en palabras de 32 bits al detectar un flanco de subida.

Salida "cif": Refleja el valor de "clk" una vez que la máquina haya concluido la iniciación del cifrador.

8. PRUEBAS Y RESULTADOS

8.1 PRUEBA FUNCIÓN MAYORITARIA

La función mayoritaria en el algoritmo A5/1 y W7 es la encargada de habilitar los relojes a los LFSR evaluando cada uno de los bits designados en dichos registros.

- Objetivo:

Comprobar el correcto funcionamiento de la función mayoritaria fundamental para el desplazamiento de los LFSR.

- Recursos:

Software: Quartus II 13.0.1 Web Edition.

- Pasos:

Se diseña en VHDL la entidad que obedece a la ecuación $F(B_{R1}, B_{R2}, B_{R3}) = B_{R1}B_{R2} \oplus B_{R1}B_{R3} \oplus B_{R2}B_{R3}$, la cual indica a cada LFSR si se debe desplazar o no.

- Resultados esperados:

Se espera dar paso de reloj a aquellos LFSR que cumplan con la función mayoritaria. De no estar habilitada la función mayoritaria ($E = 0$), simplemente permite el paso de reloj para todos. Recuerde que el paso de reloj equivale a desplazar el registro.

Tabla 7. Función mayoritaria de A5/1.

E	B_{R1}	B_{R2}	B_{R3}	F	Desplazamiento		
0	0	0	0	0	$R1$	$R2$	$R3$
0	0	0	1	0	$R1$	$R2$	
0	0	1	0	0	$R1$		$R3$
0	0	1	1	1		$R2$	$R3$
0	1	0	0	0		$R2$	$R3$
0	1	0	1	1	$R1$		$R3$
0	1	1	0	1	$R1$	$R2$	
0	1	1	1	1	$R1$	$R2$	$R3$
1	x	x	x	x	$R1$	$R2$	$R3$

- Resultados obtenidos:

Las salidas obtenidas concuerdan con los resultados esperados. En la figura 36, se observa la simulación funcional del bloque. La entrada "enai", decide si debe haber reloj regular o irregular, de estar activo (uno lógico), para todo el vector de "clks" se debe reflejar el reloj de entrada "iclk", de lo contario evalúa la entrada "taps" y solo refleja el reloj de entrada para aquellos que sean mayoría, por ejemplo si la entrada "taps" tiene valor "100" la salida es "011" ya que los ceros son mayoría.

Figura 36. Simulación funcional de la función mayoritaria.

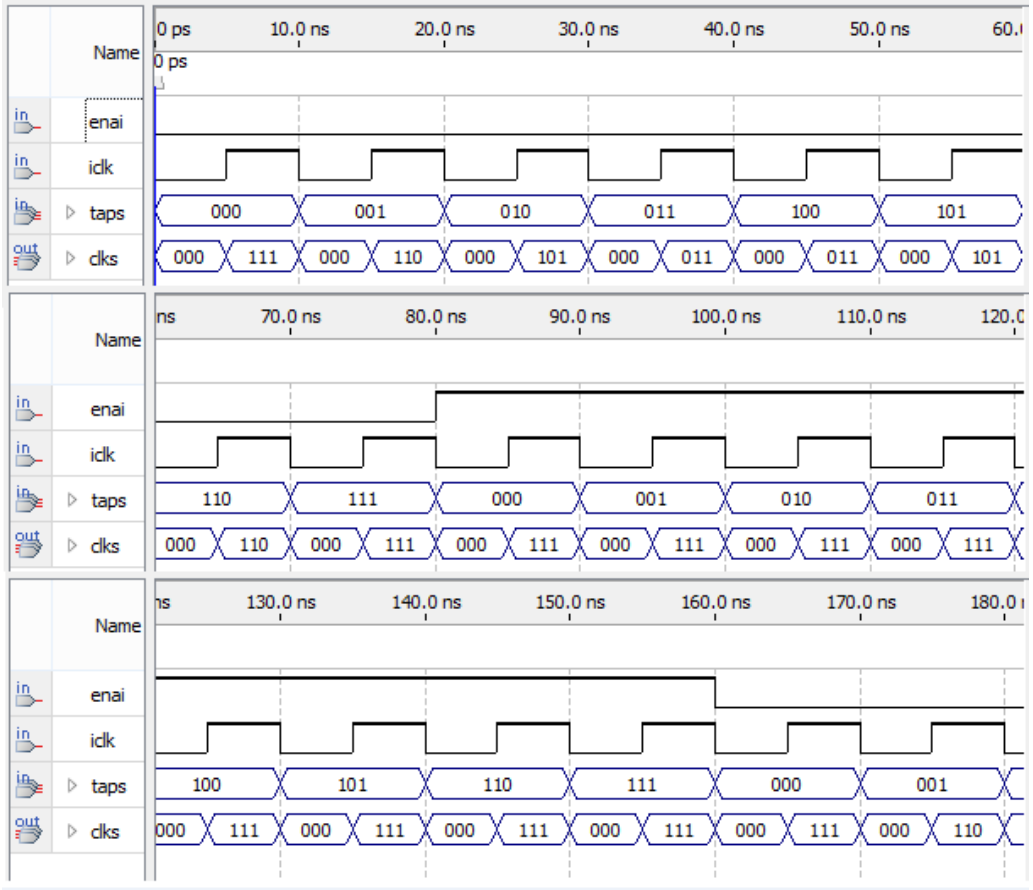
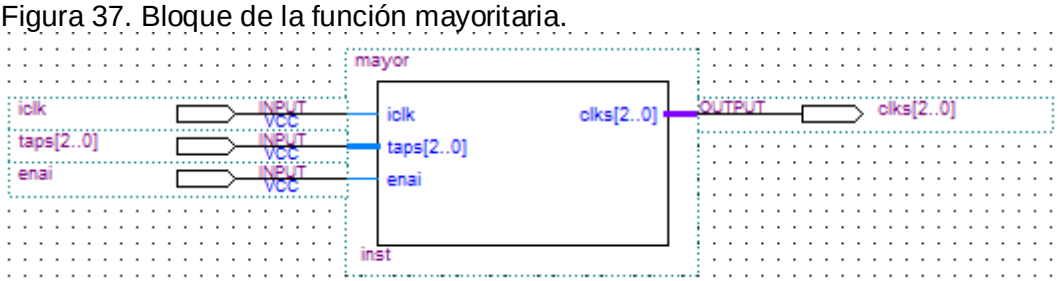


Figura 37. Bloque de la función mayoritaria.



8.2 PRUEBA LFSR PARA A5/1

El LFSR es el corazón del algoritmo A5/1, es el encargado de generar secuencias pseudo-aleatorias para el cifrado de datos. Por cada ciclo de reloj genera un bit.

- Objetivo:

Comprobar la funcionalidad del LFSR con y sin entrada de datos.

- Recursos:

Software: Quartus II 13.0.1 Web Edition y Excel 2013.

- Pasos:

Diseñar en VHDL una entidad con parámetros genéricos para su reutilización. Los parámetros son:

Longitud del LFSR: Indica la longitud en bits del registro

Habilitador de reloj: Indica el bit que alimentará la función mayoritaria.

Bits de retroalimentación: Indica que posiciones del LFSR deben operarse en OR exclusiva, el resultado retroalimentará al registro.

- Resultados esperados:

Se toma de referencia los siguientes parámetros de uno de los LFSR del algoritmo A5/1 y las claves:

Longitud del LFSR: 19.

Habilitador de reloj: 8.

Bits de retroalimentación: 18, 17, 16 y 13.

Clave privada: 4E2F4D7C1EB88B3A.

Número de trama: 3AB3CB.

Al terminar de cargar la clave y el número de trama, es decir después de 86 ciclos, el estado del LFSR es 3BEBB (Ver Excel en Anexo B).

Al cerrar el paso de datos y aplicar los desplazamientos se obtienen los siguientes resultados consignados en la tabla 8:

Tabla 8. Estados del LFSR con reloj regular.

Ciclos	Valor del estado del LFSR																				LSB
	Hexadecimal	Binario																			
0	3BEBB	0	1	1	1	0	1	1	1	1	1	0	1	0	1	1	1	0	1	1	1
1	77D77	1	1	1	0	1	1	1	1	1	0	1	0	1	1	1	0	1	1	1	0
2	6FAEE	1	1	0	1	1	1	1	1	0	1	0	1	1	1	0	1	1	1	0	1
3	5F5DD	1	0	1	1	1	1	1	0	1	0	1	1	1	0	1	1	1	0	1	1
4	3EBBB	0	1	1	1	1	1	0	1	0	1	1	1	0	1	1	1	0	1	1	1
5	7D777	1	1	1	1	1	0	1	0	1	1	1	0	1	1	1	0	1	1	1	1
6	7AEEF	1	1	1	1	0	1	0	1	1	1	0	1	1	1	0	1	1	1	1	0
7	75DDE	1	1	1	0	1	0	1	1	1	0	1	1	1	0	1	1	1	1	0	1
8	6BBBD	1	1	0	1	0	1	1	1	0	1	1	1	0	1	1	1	1	0	1	1
9	5777B	1	0	1	0	1	1	1	0	1	1	1	0	1	1	1	1	0	1	1	1
10	2EEF7	0	1	0	1	1	1	0	1	1	1	0	1	1	1	1	0	1	1	1	0
11	5DDEE	1	0	1	1	1	0	1	1	1	0	1	1	1	1	0	1	1	1	0	0
12	3BBDC	0	1	1	1	0	1	1	1	0	1	1	1	1	0	1	1	1	0	0	1

El correcto funcionamiento con un LFSR, garantiza que los demás trabajen de la misma manera, ya que solo se cambia los parámetros en el código VHDL, por lo tanto los resultados esperados para los otros LFSR después de los 86 bits debe ser (Ver Excel en Anexo B):

LFSR (22 bits) = 28A47C.

LFSR (23 bits) = 5A15B8.

- Resultados obtenidos:

Se comprobó el correcto funcionamiento del LFSR con y sin entrada de datos. En la figura 38 se observa el estado final después de haber ingresado los 86 bits de clave (enainp = 1), después de haber ingresado se inhabilita el paso de datos (enainp =0), en la figura 39 se observa los valores de estado del LFSR.

Figura 38. Simulación funcional del LFSR (19 bits) con paso de datos.

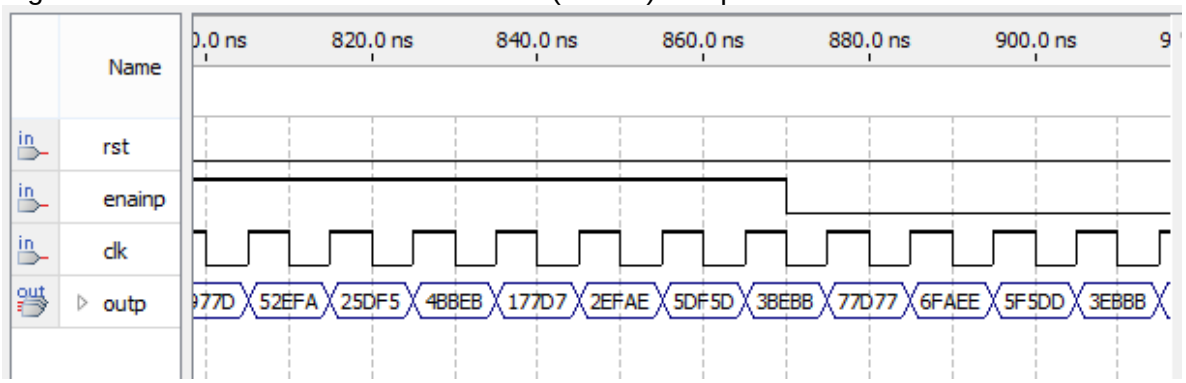
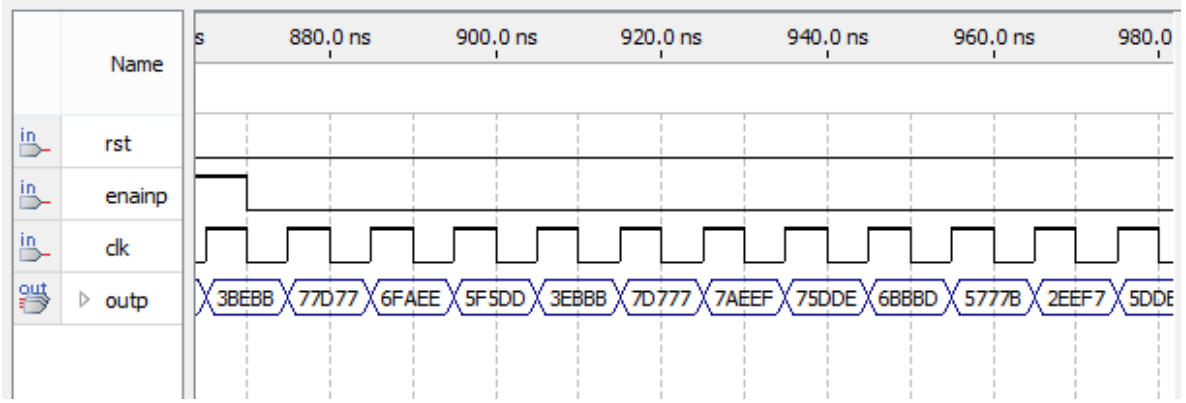


Figura 39. Simulación funcional del LFSR (19 bits) sin paso de datos.



Al igual para los registros restantes, el resultado es el esperado (Ver figura 40 y 41).

Figura 40. Simulación funcional del LFSR (22 bits).

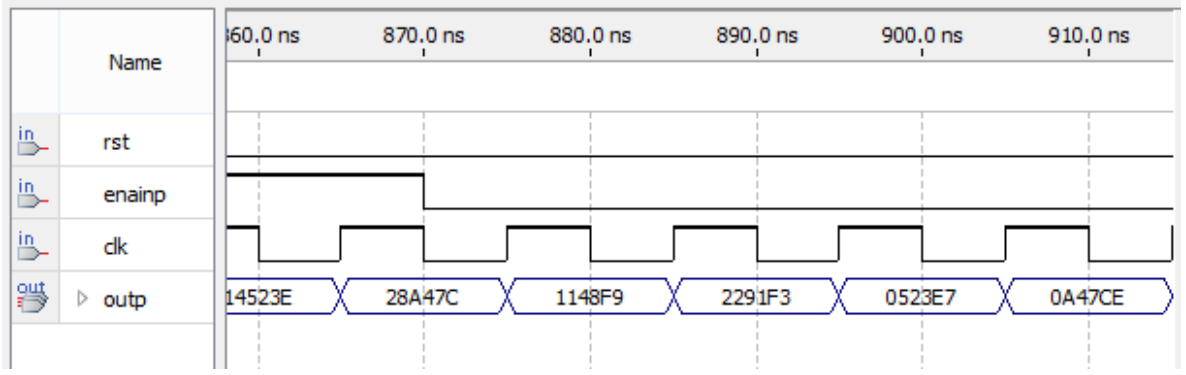


Figura 41. Simulación funcional del LFSR (23 bits)

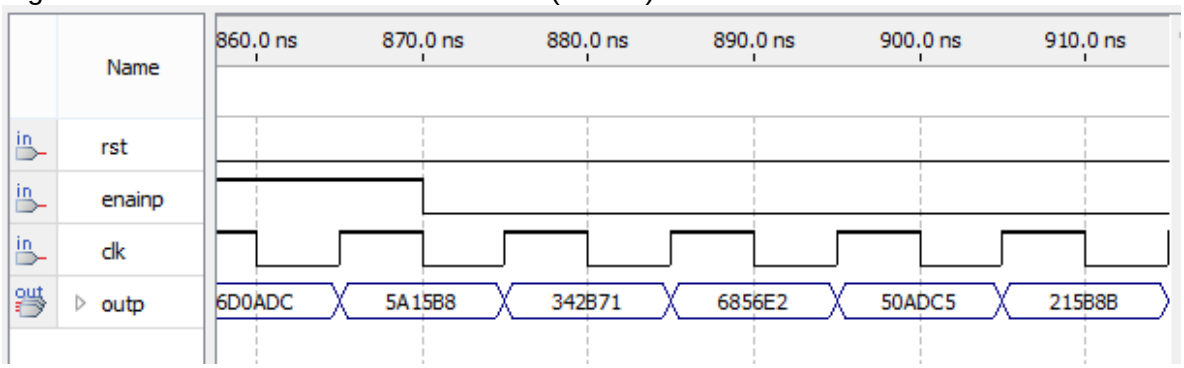
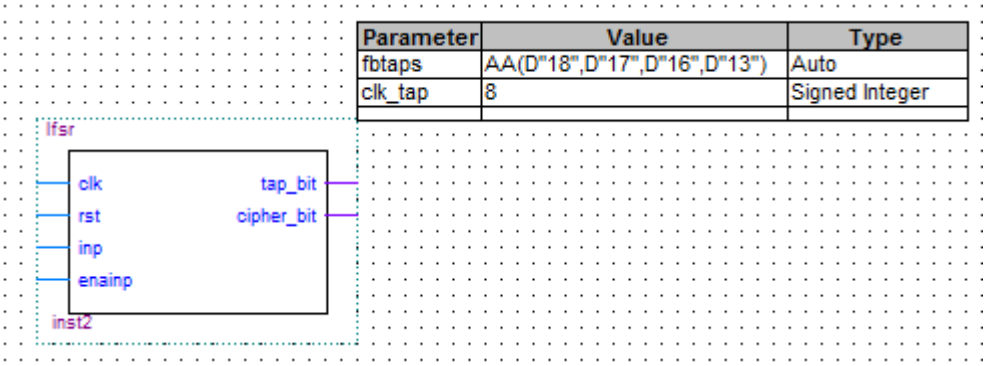


Figura 42. Bloque de LFSR genérico



8.3 PRUEBA CELDA PARA A5/1

- Objetivo:

Comprobar el correcto funcionamiento de los LFSRs y la función mayoritaria de forma conjunta. Además incluir una operación XOR entre las salidas de los LFSR, que se define como bit de cifrado.

- Recursos:

Software: Quartus II 13.0.1 Web Edition y Excel 2013.

- Pasos:

Diseñar en VHDL una entidad que una las entidades de los LFSR y la función mayoritaria, para así poder generar el bit de cifrado.

Como el reloj regular solo ocurre cuando se habilita el paso de datos, se hace eliminación de un puerto en la celda, es decir, cuando se habilite el paso, se habilita el reloj regular.

La celda debe operar debidamente cuando transicione del paso de datos al no paso, lo que también se refleja en el cambio de relojes.

- Resultados esperados:

Después de cargar los 86 bits se espera los siguientes estados de los LFSR (Ver Excel en Anexo B):

LFSR (19 bits) = 3BEBB.

LFSR (22 bits) = 28A47C.

LFSR (23 bits) = 5A15B8.

Donde

Clave privada: 4E2F4D7C1EB88B3A.

Número de trama: 3AB3CB.

A partir de este momento se empieza a considerar los bits habilitadores de reloj de cada registro y se habilita la función mayoritaria. Parte de la secuencia que se entrega es:

LFSR (19 bits) = 3BEBB 77D77 6FAEE 3EBBB 3EBBB.

LFSR (22 bits) = 1148F9 2291F3 2291F3 0523E7 0523E7.

LFSR (23 bits) = 342B71 6856E2 50ADC5 50ADC5 215B8B.

El bit de cifrado es la operación OR exclusiva entre cada una de las salidas de los LFSR, que viene siendo el bit más significativo de cada registro.

Con las secuencias anteriores se tiene la secuencia de cifrado:

Secuencia de cifrado: 011001.

- Resultados obtenidos:

En la figura 43 se ilustra el valor de los estados de cada LFSR. pin_nan es el resultado de la operación OR exclusiva entre cada una de las salidas de los LFSR. Observe que de estar desactivado "clkcell" no todos los estados siempre cambia conforme al reloj "clk", es decir que la celda está operando con reloj irregular el cual lo determina la función mayoritaria.

Figura 43. Estados del LFSR

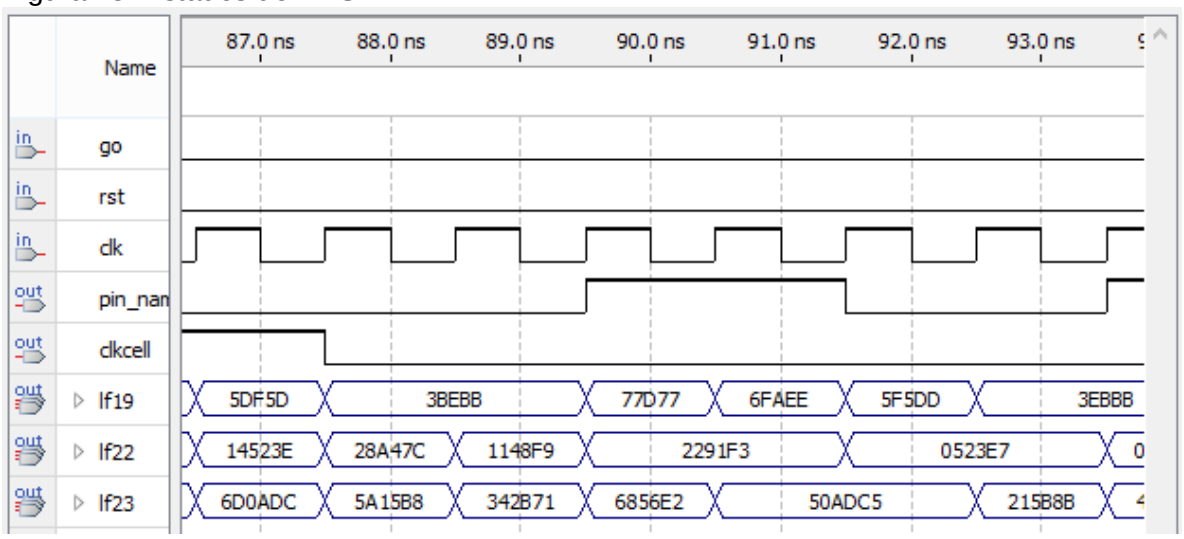
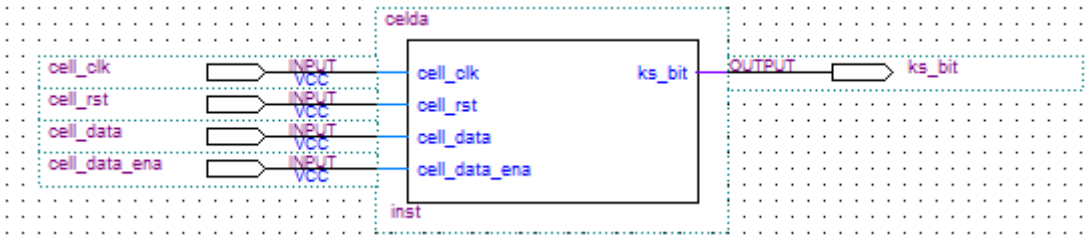


Figura 44. Bloque de celda.



8.4 PRUEBA MEMORIA DE CLAVES PARA A5/1

El algoritmo A5/1 utiliza una clave de 86 bits para realizar el cifrado, 64 bits son una clave privada y 22 bits representa el número de trama, como el algoritmo carga bit a bit esta clave, se diseña una entidad que carga estos valores y envía de forma serial.

- Objetivo:

Comprobar el debido funcionamiento de la entidad que se encarga de la lectura de la clave privada y del número de trama, y de enviar estos datos de forma serial.

- Recursos:

Software: Quartus II 13.0.1 Web Edition.

- Pasos:

Se diseñó en VHDL una entidad que lee la clave y el número de trama y envía de forma serial. Al terminar enviar la trama, incrementa está en uno.

- Resultados esperados:

Teniendo como valores:

Clave privada = 4E2F4D7C1EB88B3A.

Número de trama = 3AB3CB.

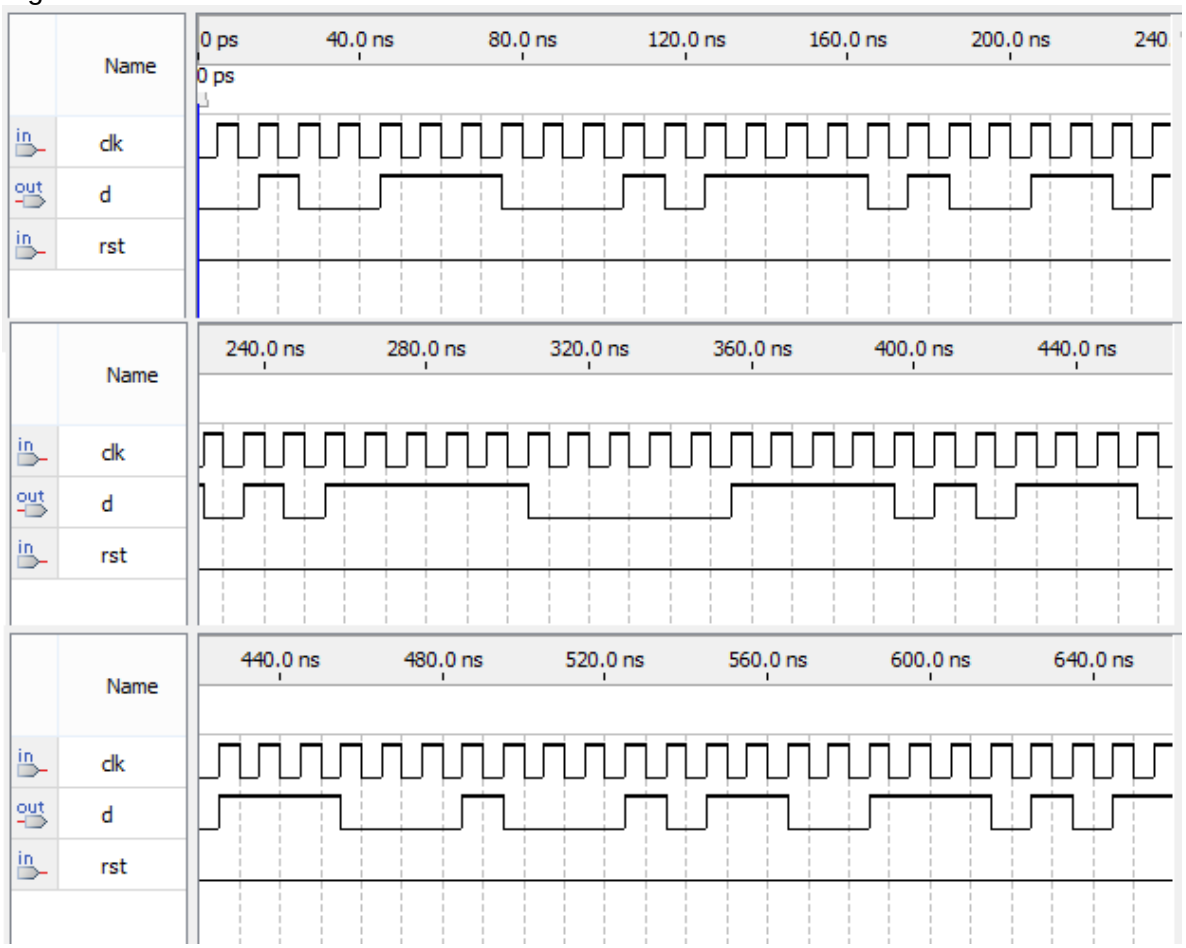
Se espera enviar estos mismos valores de forma serial, empezando por el bit más significativo.

- Resultados obtenidos:

En la simulación de la figura 45, se ilustra él envió serial de los datos.

Clave privada en secuencia de bits = 0100 1110 0010 111 0100 1101 0111 1100 0001 1110 1011 1000 1000 1011 0011 1010.

Figura 45. Simulación funcional de la memoria de claves.



8.5 PRUEBA MEMORIA DE CIFRADO PARA A5/1

El algoritmo A5/1 es diseñado para cifrar tramas de 228 bit. La memoria guarda 228 bits que son los correspondientes para cifrar el texto plano.

- Objetivo:

Comprobar el correcto funcionamiento del almacenamiento de la secuencia de cifrado, recibe 228 bits de forma serial y guarda en un registro del mismo tamaño.

- Recursos:

Software: Quartus II 13.0.1 Web Edition.

- Pasos:

Diseñar en VHDL la entidad encargada de capturar los datos seriales y guardarlos en un registro de la misma longitud de la trama.

- Resultados esperados:

De una secuencia de bits leída, reflejarla en el registro. Teniendo como referencia el siguiente vector:

Secuencia de datos (bits) = 4 E2F4 D7C1 EB88 B3AE ACF2 D38B D35F 07AE 22CE BAB3 CC4E 2F4D 7C1E B88B.

El vector anterior debe ser idénticamente al generado por el bloque de "memoria de cifrado", no idéntico por tramas de 86 bits, ya que al terminar él envió de una, aumenta el número de trama en uno, solo la clave se mantiene idéntica.

- Resultados obtenidos:

La salida obtenida acorde a la simulación fue:

Salida = 4 E2F4 D7C1 EB88 B3AE ACF2 D38B D35F 07AE 22CE BAB3 CC4E 2F4D 7C1E B88B (Ver figura 46).

Figura 46. Simulación funcional de la memoria de cifrado.

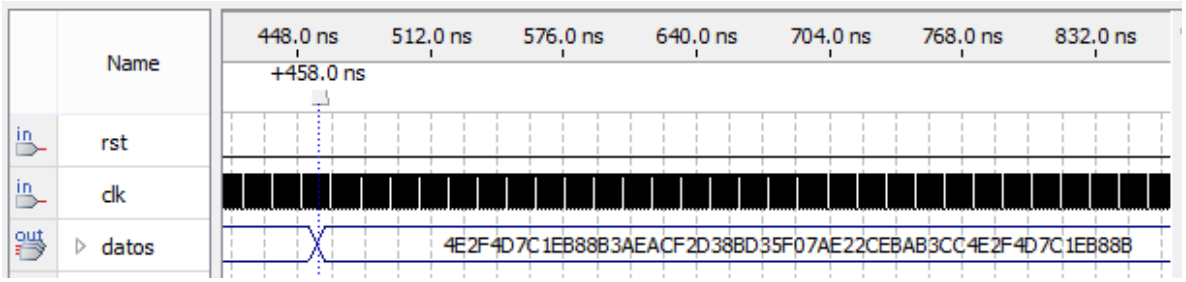
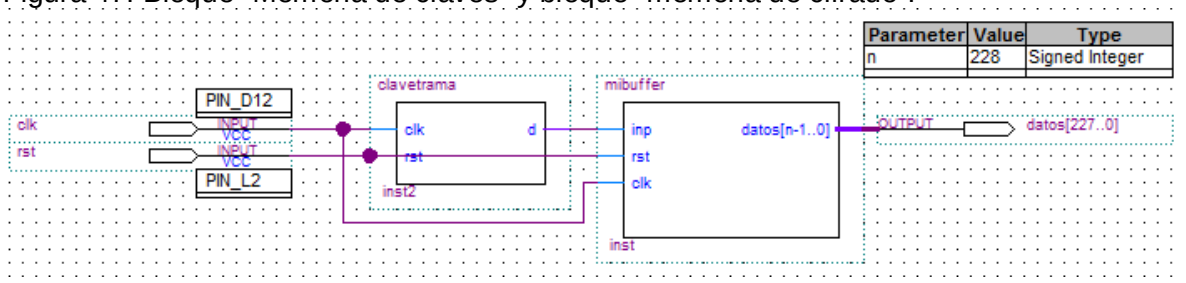


Figura 47. Bloque "Memoria de claves" y bloque "memoria de cifrado".



8.6 PRUEBA MAQUINA DE ESTADOS Y CIFRADOR A5/1

- Objetivo:

Controlar los pasos del algoritmo A5/1 y comprobar el correcto funcionamiento de cifrador al interconectar la máquina de estados con las demás entidades diseñadas en VHDL.

- Recursos:

Software: Quartus II 13.0.1 Web Edition y Excel 2013.

- Pasos:

Diseñar en VHDL una máquina de 4 estados que genere los relojes de salida.

- Resultados esperados:

Se espera que la maquina inicie en 0 y culmine al generar la secuencia de cifrado. De ser correcto generará la secuencia de cifrado de 228 bits correctamente.

Clave privada: = 4E2F4D7C1EB88B3A.

Número de trama = 3AB3CB.

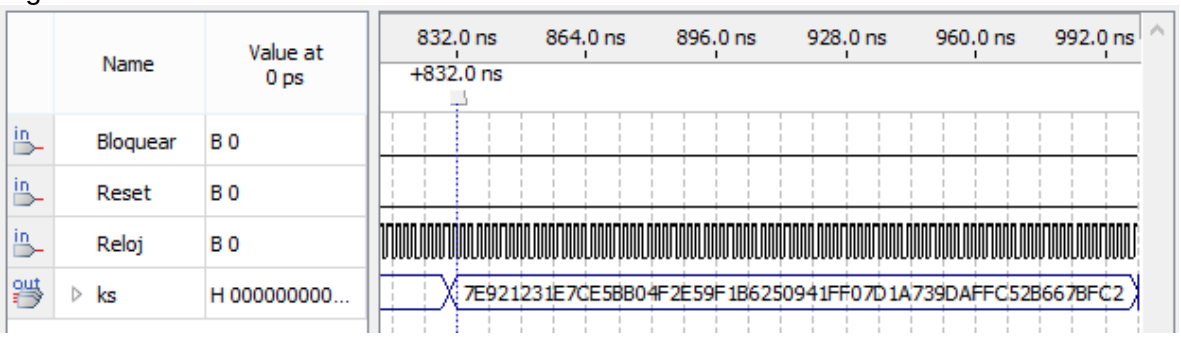
Secuencia de cifrado = 7 E921 231E 7CE5 BB04 F2E5 9F16 2509 41FF 07DD 1A73 9DAF FC52 B667 BFC2.

Los resultados anteriores son tomados del archivo Excel (Anexo B).

- Resultados obtenidos:

La secuencia de cifrado obtenida es = 7 E921 231E 7CE5 BB04 F2E5 9F16 2509 41FF 07DD 1A73 9DAF FC52 B667 BFC2. (Ver figura 48)

Figura 48. Simulación funcional del cifrador A5/1

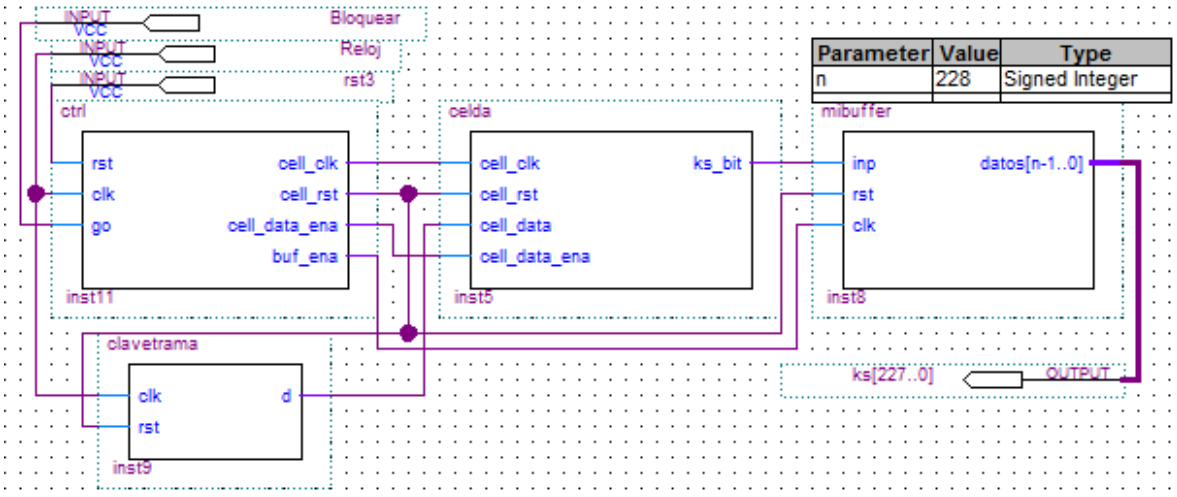


- Conclusión:

El cifrador opera correctamente, por lo tanto no hay desincronización entre las entidades. El cifrador culmina en 416 ciclos de máquina, lo que resume que por bit cifrado gasta 1,86 ciclos de máquina.

El descifrador es el mismo cifrador, ya que no hay dependencia del texto plano, simplemente durante el envío tanto como emisor y receptor tiene que coincidir en la clave y el número de trama.

Figura 49. Cifrador / Descifrador A5/1.



8.7 PRUEBA LFSR PARA W7

- Objetivo:

Comprobar la funcionalidad del LFSR para W7.

- Recursos:

Software: Quartus II 13.0.1 Web Edition y Excel 2013.

- Pasos:

Diseñar en VHDL una entidad con parámetros genéricos para su reutilización. Los parámetros son:

Longitud del LFSR: Indica la longitud en bits del registro

Habilitador de reloj: Indica el bit que alimentará la función mayoritaria.

Bits de retroalimentación: Indica que posiciones del LFSR deben operarse en OR exclusiva, el resultado retroalimentará al registro.

Bits de salida filtrada: Indica que conjuntos de posiciones del LFSR deben operarse con AND, cada resultado de los conjuntos, junto con el bit más significativo del LFSR se opera con OR exclusiva.

- Resultados esperados:

A diferencia del LFSR del algoritmo A5/1, este no carga la clave de forma serial, en cambio su estado inicial es la clave de 128 bits que carga paralelamente dentro del registro.

Su funcionamiento debe ser el mismo al planteado en el algoritmo A5/1 con la excepción anterior y su salida bit filtrada.

Se tiene como parámetros:

Valor inicial del registro (38 bits) = 0CB8FB5A7E.

Bits de retroalimentación = 37,32,29,27,26,21,20,14,12,11,10,9,8,5,2,0.

Bits de salida filtrada = (36, 33,0), (32, 29,0), (28, 25,22). Los bits dentro de los paréntesis se operan con AND, y entre estos con OR exclusiva.

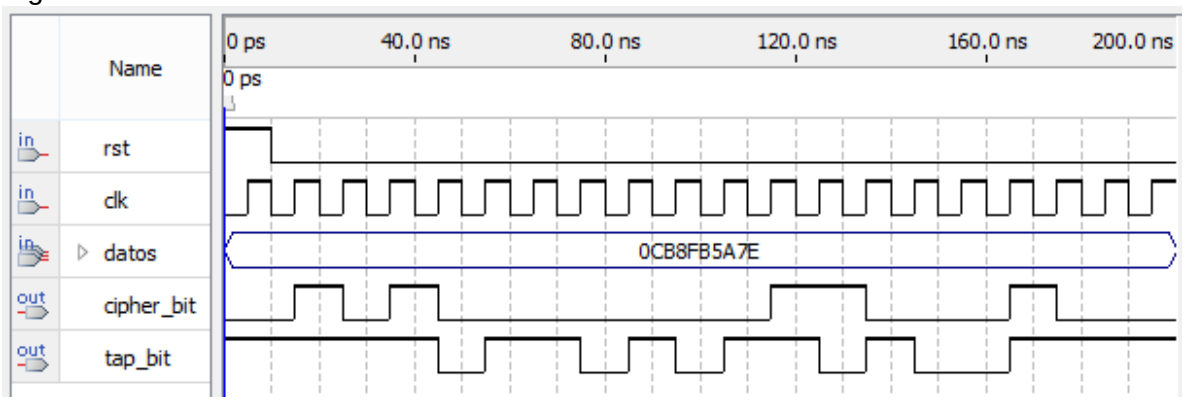
La hoja de cálculo de Excel arroja la siguiente secuencia de bits de salida:

Salida de bit filtrado = 0101 0000 0001 1000.

- Resultados obtenidos:

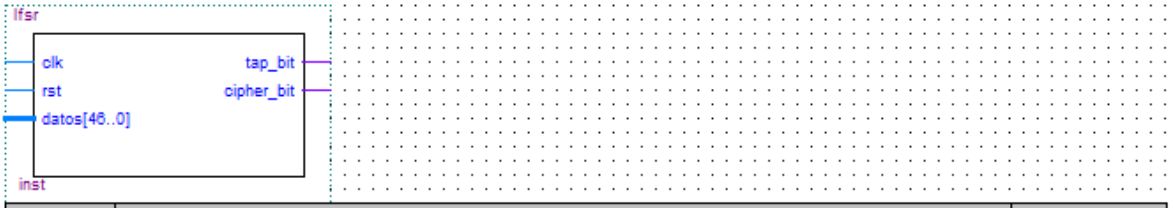
Se obtuvo la secuencia de bits de salida: 0101 0000 0001 1000 (Ver figura 50).

Figura 50. Simulación funcional del LFSR de W7.



"cipher_bit" es la salida de bit filtrado del LFSR.

Figura 51. LFSR de W7.



Parameter	Value	Type
fbtaps	AA(D"37",D"32",D"29",D"27",D"26",D"21",D"20",D"14",D"12",D"11",D"10",D"9",D"8",D"5",D"2",D"0")	Auto
clk_tap	22	Signed Integer
filter_taps	AA(A(D"36",D"33",D"0"),A(D"32",D"29",D"0"),A(D"28",D"25",D"22"))	Auto

8.8 PRUEBA CELDA PARA W7

- Objetivo:

Juntar los LFSR y la función mayoritaria, el cual es requerido por el algoritmo.

- Recursos:

Software: Quartus II 13.0.1 Web Edition.

- Pasos:

Diseñar en VHDL una entidad con parámetros genéricos para su reutilización. Los parámetros son los mismos especificados en la entidad LFSR para W7, con la excepción de que se requieren tres veces, al tener tres LFSR.

- Resultados esperados:

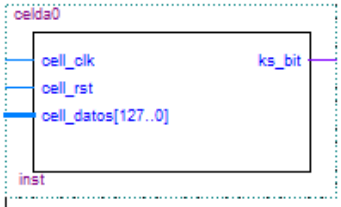
Al componerse el algoritmo de ocho celdas, conjuntamente debe generar un byte de cifrado.

De ser reiniciada la celda, toma como valor inicial la clave privada, que ingresa por su puerto.

- Conclusión:

Esta celda al ser análoga a la celda del algoritmo A5/1 debe operar correctamente, no hay necesidad de realizar pruebas por cada una ya que operan paralelamente, el correcto funcionamiento de una sola garantiza el correcto funcionamiento del total de las celdas, de no serlo, es porque no se obedece a los parámetros de construcción establecidos por el algoritmo. En la figura 52, se ilustra una celda que genera el bit menos significativo acorde a los parámetros establecidos, estos se declaran en los parámetros genéricos de la entidad.

Figura 52. LFSR de W7.



Parameter	Value	Type
la0fbtaps	AA(D"37",D"32",D"29",D"27",D"26",D"21",D"20",D"14",D"12",D"11",D"10",D"9",D"8",D"5",D"2",D"0")	Auto
la0clk_tap	22	Signed Integer
la0filter_taps	AA(A(D"36",D"33",D"0"),A(D"32",D"29",D"0"),A(D"28",D"25",D"22"))	Auto
lb0fbtaps	AA(D"42",D"5",D"3",D"2")	Auto
lb0clk_tap	25	Signed Integer
lb0filter_taps	AA(A(D"41",D"39",D"0"),A(D"38",D"36",D"0"),A(D"35",D"33",D"31"))	Auto
lc0fbtaps	AA(D"46",D"4")	Auto
lc0clk_tap	27	Signed Integer
lc0filter_taps	AA(A(D"45",D"40",D"0"),A(D"39",D"34",D"0"),A(D"33",D"28",D"23"))	Auto

8.9 PRUEBA MAQUINA DE ESTADOS Y CIFRADOR W7

- Objetivo:

Comprobar el correcto funcionamiento del cifrador al unir la máquina de estados con los demás entidades requeridas por el algoritmo.

- Recursos:

Software: Quartus II 13.0.1 Web Edition.

- Pasos:

Se diseñó en VHDL una máquina de estados, que es la unidad de control del cifrador. Cuenta con tres estados, el primero lee la clave, el segundo inicializa el cifrador y el tercero cifra los datos.

- Resultados esperados (Thomas & Anthony, 2002):

Con los siguientes valores:

Clave = 0001 0203 0405 0607 0809 0A0B 0C0D 0E0F

Tabla 9. Vector de prueba en texto claro.

00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F
10	11	12	13	14	15	16	17	18	19	1A	1B	1C	1D	1E	1F
20	21	22	23	24	25	26	27	28	29	2A	2B	2C	2D	2E	2F
30	31	32	33	34	35	36	37	38	39	3A	3B	3C	3D	3E	3F
40	41	42	43	44	45	46	47	48	49	4A	4B	4C	4D	4E	4F
50	51	52	53	54	55	56	57	58	59	5A	5B	5C	5D	5E	5F
60	61	62	63	64	65	66	67	68	69	6A	6B	6C	6D	6E	6F
70	71	72	73	74	75	76	77	78	79	7A	7B	7C	7D	7E	7F
80	81	82	83	84	85	86	87	88	89	8A	8B	8C	8D	8E	8F
90	91	92	93	94	95	96	97	98	99	9A	9B	9C	9D	9E	9F
A0	A1	A2	A3	A4	A5	A6	A7	A8	A9	AA	AB	AC	AD	AE	AF
B0	B1	B2	B3	B4	B5	B6	B7	B8	B9	BA	BB	BC	BD	BE	BF
C0	C1	C2	C3	C4	C5	C6	C7	C8	C9	CA	CB	CC	CD	CE	CF
D0	D1	D2	D3	D4	D5	D6	D7	D8	D9	DA	DB	DC	DD	DE	DF
E0	E1	E2	E3	E4	E5	E6	E7	E8	E9	EA	EB	EC	ED	EE	EF
F0	F1	F2	F3	F4	F5	F6	F7	F8	F9	FA	FB	FC	FD	FE	FF

(Thomas & Anthony, 2002)

Se espera el siguiente texto cifrado:

Tabla 10. Texto claro cifrado.

B6	FD	65	03	38	08	B6	0A	6F	47	10	CF	DA	FB	1F	7E
71	66	40	5E	38	1D	26	07	D4	7D	58	07	5F	B7	93	83
EC	28	FC	B7	45	7E	56	EF	3D	32	12	57	F0	6C	9D	85
A6	11	66	02	1A	BC	51	B3	D8	AD	11	0D	A3	FE	ED	F0
D6	E4	69	7C	3F	24	9F	E1	14	9E	87	E3	9B	11	5A	E1
B4	AE	27	74	BE	66	02	A1	59	F8	EA	70	04	39	DA	54
0A	F3	B1	13	9D	3A	99	D6	9C	18	18	3C	32	EA	7F	72
89	68	4A	D4	77	4E	F8	25	9D	94	8C	B4	6F	50	F2	33
2B	F4	72	AA	1B	62	EC	F3	64	B0	6F	87	07	B0	31	7E
EC	11	C0	7E	82	6B	60	7C	17	73	A1	71	1E	7F	2D	C7
6C	E6	35	9C	DC	1F	0B	55	18	C8	32	08	19	0C	19	03
8B	FE	54	8D	A3	C8	69	FD	5E	09	96	53	66	D3	1B	6A
F2	9D	D0	96	02	DF	9A	BE	72	42	59	1E	CA	D1	A2	33
AE	EA	65	4D	D1	BB	76	CC	13	70	E1	37	0D	F4	F2	23
8E	A9	09	71	89	4C	16	C1	DC	B0	9F	5E	94	68	AA	0F
1F	D9	0C	70	F5	02	CD	67	17	6C	14	78	0B	0E	6B	1B

(Thomas & Anthony, 2002)

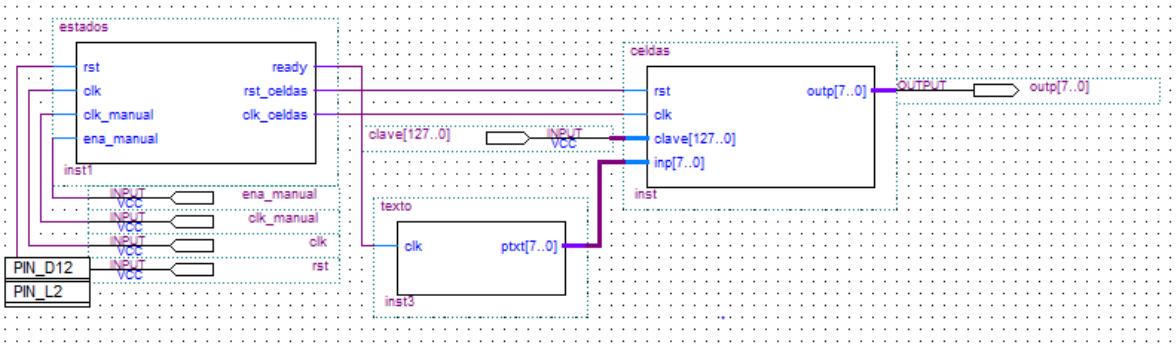
- Resultados obtenidos:

El resultado obtenido es el mismo al esperado. Del texto plano de referencia y la clave se obtiene el texto cifrado correspondiente (Ver Anexo A).

- Conclusión:

El cifrador concluye su inicialización en 1032 ciclos de máquina, el autor del algoritmo recomienda no usar la misma clave para cifrar más de 2^46 bytes. Si se toma este límite como el tamaño de la secuencia a cifrar, el gasto de ciclos por bit debido a la iniciación del cifrador es despreciable, aproximadamente de 1.83x10^-12 ciclos.

Figura 53. Cifrador / Descifrador W7.



8.10 PRUEBA BLOQUE HÉLIX Y GENERADOR DE CLAVE DE TRABAJO

- Objetivo:

Comprobar el correcto funcionamiento del bloque de hélix, y del generador de la clave de trabajo.

- Recursos:

Software: Quartus II 13.0.1 Web Edition.

- Pasos:

Diseñar en VHDL el bloque Hélix, el cual el algoritmo genera los datos pseudo-aleatorios.

Una vez creada aplicar los pasos necesarios para crear la clave de trabajo, la cual solo es generada por este bloque y solo depende de este. Para este se diseña en VHDL una función que haga uso de este bloque y retorne la clave trabajo.

Para crear la clave de trabajo, las entradas de 32 bits P_i , $x_{i,0}$, $x_{i,1}$ en el bloque Hélix, siempre tienen como valor cero.

Y se realiza los siguientes pasos:

1. Se carga la clave en las palabras $K_{32},...,K_{39}$.

2. Las palabras claves $K_0,...,K_7$ forman la clave trabajo, la cual se define con la ecuación 4.

- Resultados esperados:

Con el vector de prueba:

Clave privada = 48 65 6C 69 78 (Primer byte menos significativo).

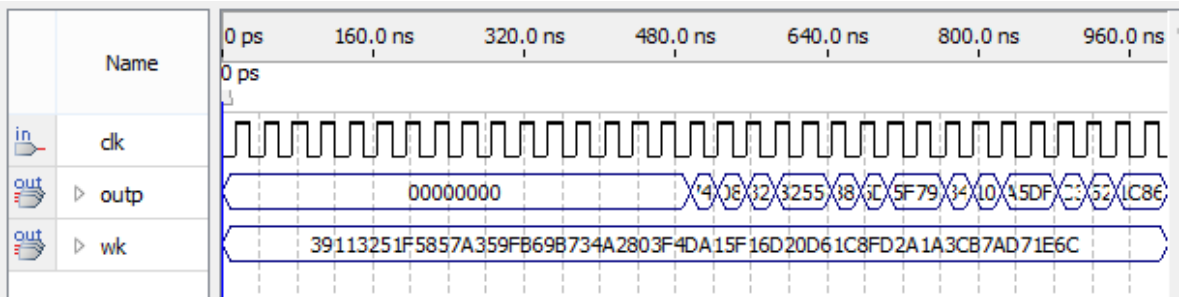
Se espera como que la clave de trabajo generada sea:

Clave de trabajo = 6E E9 A7 6C BD 0B F6 20 A6 D9 B7 59 49 D3 39 95 04 F8 4A D6 83 12 F9 06 ED D1 A6 98 9E C8 9D 45 (Primer byte menos significativo) (Thomas & Anthony, 2002).

- Resultados obtenidos:

Clave de trabajo = 6E E9 A7 6C BD 0B F6 20 A6 D9 B7 59 49 D3 39 95 04 F8 4A D6 83 12 F9 06 ED D1 A6 98 9E C8 9D 45 (Ver simulación de la figura 54).

Figura 54. Clave de trabajo generada.



8.11 PRUEBA EXTENSIÓN DE NONCE DE HÉLIX

- Objetivo:

Comprobar el correcto de funcionamiento de la extensión del nonce, el cual es requerido para la generación de la palabra clave $x_{i,1}$.

- Recursos:

Software: Quartus II 13.0.1 Web Edition.

- Pasos:

En VHDL se diseña una función que se encarga de extender el nonce con 4 palabras más.

- Resultados esperados:

Teniendo el nonce como:

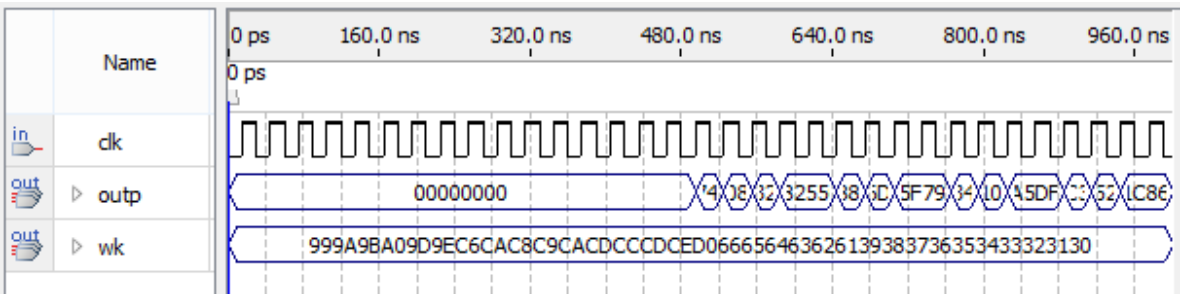
$N_0,...,N_3 = 30313233,34353637,38396162,63646566$ (Thomas & Anthony, 2002).

Las cuatro palabras que se añaden al nonce, debe obedecer a la ecuación $N_k := (k \bmod 4) - N_{k-4} (\bmod 2^{32})$ para $K = 4...,7$.

- Resultados obtenidos:

Se obtienen 8 palabras dobles (ver figura 55) que obedecen a la ecuación 5.

Figura 55. Simulación funcional de la extensión del nonce.



8.12 PRUEBA CIFRADOR HÉLIX

- Objetivo:

Comprobar el correcto funcionamiento del cifrador, con el diseño de la máquina de estados y las funciones que generan las palabras claves para el cifrador.

- Recursos:

Software: Quartus II 13.0.1 Web Edition.

- Pasos:

Se crean funciones en VHDL para generar las palabras claves y una máquina de estados que inicializa el cifrador y permite el cifrado de palabras de texto plano (32 bits).

Se diseña la máquina de estados, con dos estados. El primero inicializa el cifrador, el cual se demora el mismo número de ciclos de reloj como el número de veces en que se utiliza el bloque Hélix. El segundo estado por ciclo de reloj cifra el texto plano por palabras de 32 bits.

- Resultados esperados:

Teniendo como entrada los vectores de prueba: (Primer byte menos significativo)

Clave privada= 48 65 6c 69 78.

Nonce = 30 31 32 33 34 35 36 37 38 39 61 62 63 64 65 66.

Texto plano = 48 65 6c 6c 6f 2c 20 77 6f 72 6c 64 21.

Se genera las siguientes salidas = 6c 1e d7 7a cb a3 a1 d2 8f 1c d6 20 6d f1 15 da f4 03 28 4a 73 9b b6 9f 35 7a 85 f5 51 32 11 39.

Clave de trabajo = 6c 1e d7 7a cb a3 a1 d2 8f 1c d6 20 6d f1 15 da f4 03 28 4a 73 9b b6 9f 35 7a 85 f5 51 32 11 39.

Texto cifrado = 6c 4c 27 b9 7a 82 a0 c5 80 2c 23 f2 0d.

- Resultados obtenidos:

Se obtuvo la secuencia de cifrado: 6c 4c 27 b9 7a 82 a0 c5 80 2c 23 f2 0d (Ver figura 56).

Figura 56. Simulación funcional del cifrador Hélix.

Los bytes de cifrado demás son descartados.

- Conclusión:

El cifrador Hélix funciona correctamente, requiere de 15 ciclos de reloj para su iniciación.

8.13 RESULTADOS DE A5/1

En base a los resultados de compilación de Quartus II y del simulador, se obtienen los siguientes resultados:

Parámetros de potencia:

Tabla 11. Parámetros de potencia de A5/1.

Opciones	Escenario	Valores por Defecto
Uso de compilación inteligente	no	no
Habilitar Assembler paralelo y analizador de tiempo de TimeQuest durante la compilación	si	si
Habilitar tabla de reporte compacto	no	no
Tasa de alternación por defecto de potencia	12.5%	12.5%
Tasa de alternación por defecto de potencia de entradas y salidas	12.5%	12.5%
Uso de estimación sin vector	si	si
Uso de archivos en las entradas	no	no
Fallos de filtro en lector de archivos VCD	si	si
Señal de reporte de análisis de potencia	no	no
Características de potencia del dispositivo	normal	normal
Calcular automáticamente temperatura de unión	si	si
Temperatura especifica de unión	25.0	25.0

Temperatura de ambiente	25.0	25.0
Uso de solución de refrigeración personalizada	no	no
Temperatura de la tarjeta	25.0	25.0
Habilitar HPS	no	no
Frecuencia de procesador	0.0	0.0

Resumen de potencia:

Tabla 12. Resumen de potencia de A5/1.

Estado de flujo	**exitoso**
Versión de Quartus II 64-bits	13.0.1 estructura 232 web edition
Nombre	A5_1
Entidad principal	A5_1
Familia	Cyclone II
Dispositivo	EP2C20F484C7
Potencia térmica disipada	67.12 mW
Potencia térmica disipada en el núcleo dinámico	0.00 mW
Potencia térmica disipada en el núcleo estático	47.35 mW
Disipación térmica de potencia en las entradas y salidas	19.77 mW

Recursos de la FPGA:

Tabla 13. Recursos utilizados de la FPGA de A5/1.

Estado de Flujo	**Exitoso**
Versión de Quartus II 64-bits	13.0.1 estructura 232 web edition
Nombre	A5_1
Entidad principal	A5_1
Familia	Cyclone II
Dispositivo	EP2C20F484C7
Elementos lógicos	291 de 18,752 (2%)
Funciones combinaciones	235 de 18757 (1%)
Registros lógicos digitales	13 de 18,752 (1%)
Pines usados	4 de 315 (1%)
Pines virtuales	o
Memoria usada	0 bits de 239.616 bits (0%)
PLLs	0 de 4 (0%)

Simulación de velocidad:

Utilizando el simulador de Quartus II, se comprueba a qué velocidad y a la cual no opera correctamente el cifrador.

En la figura 57, la velocidad mínima en que el cifrador A5/1 entrega cifrado de datos correctamente y de forma segura a través del tiempo es a 1 ciclo/ 10 ns. En la figura 58 donde el cifrado no es correcto ocurre a una velocidad de 1 ciclo / 5 ns.

Figura 57. Velocidad mínima de operación del cifrador A5/1 a 1 ciclo / 10 ns.

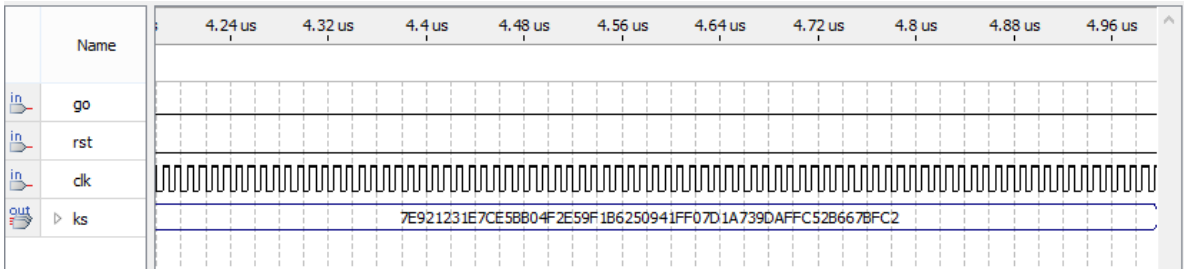
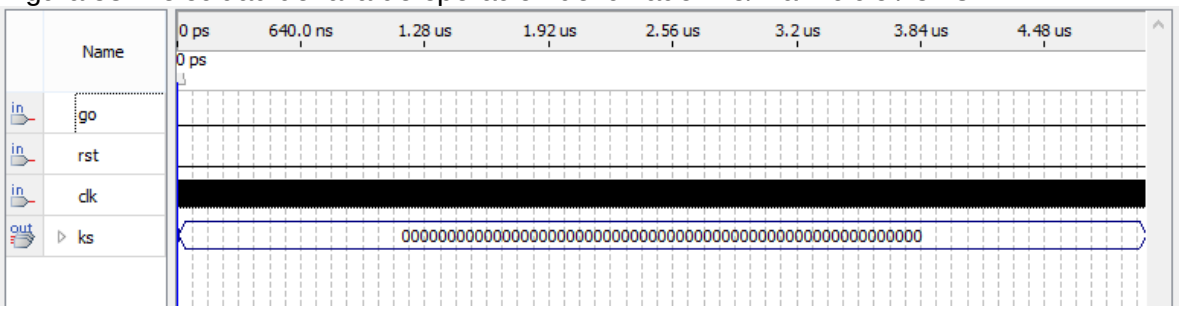


Figura 58. Velocidad de falla de operación del cifrador A5/1 a 1 ciclo / 5 ns.



8.14 RESULTADOS DE W7

De igual manera, en base a los resultados de compilación de Quartus II y del simulador, se obtienen los siguientes resultados:

Parámetros de potencia:

Tabla 14. Parámetros de potencia de W7.

Opciones	**Escenario**	**Valores por Defecto**
Uso de compilación inteligente	no	no
Habilitar Assembler paralelo y analizador de tiempo de TimeQuest durante la compilación	si	si

Habilitar tabla de reporte compacto	no	no
Tasa de alternación por defecto de potencia	12.5%	12.5%
Tasa de alternación por defecto de potencia de entradas y salidas	12.5%	12.5%
Uso de estimación sin vector	si	si
Uso de archivos en las entradas	no	no
Fallos de filtro en lector de archivos VCD	si	si
Señal de reporte de análisis de potencia	no	no
Características de potencia del dispositivo	normal	normal
Calcular automáticamente temperatura de unión	si	si
Temperatura especifica de unión	25.0	25.0
Temperatura de ambiente	25.0	25.0
Uso de solución de refrigeración personalizada	no	no
Temperatura de la tarjeta	25.0	25.0
Habilitar HPS	no	no
Frecuencia de procesador	0.0	0.0

Resumen de potencia:

Tabla 15. Resumen de potencia de W7.

estado de flujo	**exitoso**
Versión de Quartus II 64-bits	13.0.1 estructura 232 web edition
Nombre	w 7
Entidad principal	w 7
Familia	Cyclone II
Dispositivo	EP2C20F484C7
Potencia térmica disipada	83.14 mW
Potencia térmica disipada en el núcleo dinámico	0.00 mW
Potencia térmica disipada en el núcleo estático	47.38 mW
Disipación térmica de potencia en las entradas y salidas	35.76 mW

Recursos de la FPGA:

Tabla 16. Recursos utilizados de la FPGA de W7.

Estado de Flujo	**Exitoso**
Versión de Quartus II 64-bits	13.0.1 estructura 232 web edicion
Nombre	w 7

Entidad principal	w 7
Familia	Cyclone II
Dispositivo	EP2C20F484C7
Elementos lógicos	2,394 de 18,752 (13%)
Funciones combinaciones	2,392 de 18757 (13%)
Registros lógicos digitales	1,054 de 18,752 (6%)
Pines usados	104 de 315 (44%)
Pines virtuales	o
Memoria usada	0 bits de 239.616 bits (0%)
PLLs	0 de 4 (0%)

Simulación de velocidad:

Utilizando el simulador de Quartus II, se comprueba a qué velocidad y a la cual no opera correctamente el cifrador.

En la figura 59, la velocidad mínima en que el cifrador W7 entrega cifrado de datos correctamente y de forma segura a través del tiempo es de 1 ciclo/10 ns. En la figura 60 donde el cifrado no es correcto ocurre a una velocidad de 1 ciclo / 7 ns.

Figura 59. Velocidad mínima de operación del cifrador W7 a 1 ciclo / 10 ns.

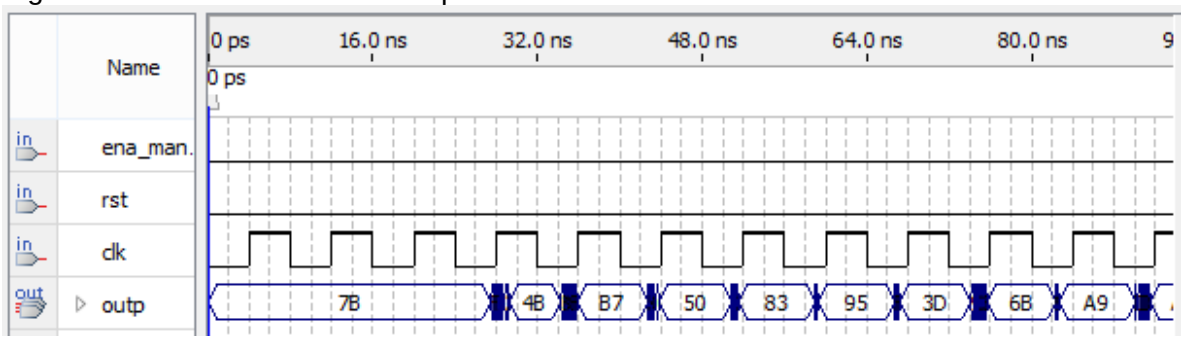
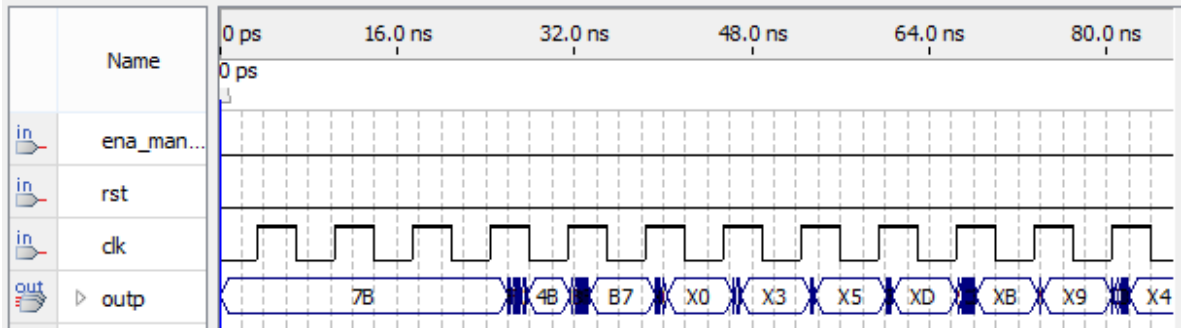


Figura 60. Velocidad de falla de operación del cifrador A5/1 a 1 ciclo / 7 ns.



8.15 RESULTADOS DE HÉLIX

En base a los resultados de compilación de Quartus II y del simulador, se obtienen los siguientes resultados:

Parámetros de potencia:

Tabla 17. Parámetros de potencia de Hélix.

Opciones	**Escenario**	**Valores por Defecto**
Uso de compilación inteligente	no	no
Habilitar Assembler paralelo y analizador de tiempo de TimeQuest durante la compilación	si	si
Habilitar tabla de reporte compacto	no	no

Tasa de alternación por defecto de potencia	12.5%	12.5%
Tasa de alternación por defecto de potencia de entradas y salidas	12.5%	12.5%
Uso de estimación (Sin Vector)	si	si
Uso de archivos en las entradas	no	no
Fallos de filtro en lector de archivos VCD	si	si
Señal de reporte de análisis de potencia	no	no
Características de potencia del dispositivo	normal	normal
Calcular automáticamente temperatura de unión	si	si
Temperatura especifica de unión	25.0	25.0
Temperatura de ambiente	25.0	25.0
Uso de solución de refrigeración personalizada	no	no
Temperatura de la tarjeta	25.0	25.0
Habilitar HPS	no	no
Frecuencia de procesador	0.0	0.0

Resumen de potencia:

Tabla 18. Resumen de potencia de Hélix.

Estado de Flujo	**Exitoso**
Versión de Quartus II 64-bits	13.0.1 estructura 232 web edición
Nombre	he_helix
Entidad principal	he_helix
Familia	Cyclone II
Dispositivo	EP2C20F484C7
Potencia térmica disipada	70.71 mW
Potencia térmica disipada en el núcleo dinámico	0.00 mW
Potencia térmica disipada en el núcleo estático	47.36 mW
Disipación térmica de potencia en las entradas y salidas	23.35 mW

Recursos de la FPGA:

Tabla 19. Recursos de la FPGA de Hélix.

Estado de Flujo	**Exitoso**
Versión de Quartus II 64-bits	13.0.1 estructura 232 web edition
Nombre	he_helix

Entidad principal	he_helix
Familia	Cyclone II
Dispositivo	EP2C20F484C7
Elementos lógicos	1,999 de 18,752 (13%)
Funciones combinaciones	1,996 de 18757 (13%)
Registros lógicos digitales	257 de 18,752 (6%)
Pines usados	257 de 315 (44%)
Pines virtuales	0
Memoria usada	0 bits de 239.616 bits (0%)
PLLs	0 de 4 (0%)

Simulación de velocidad:

Utilizando el simulador de Quartus II, se comprueba a qué velocidad y cual no opera correctamente el cifrador.

En la figura 61, la velocidad mínima en que el cifrador Hélix entrega cifrado de datos correctamente y de forma segura a través del tiempo es de 1 ciclo / 30 ns. En la figura 62 donde el cifrado no es correcto ocurre a una velocidad de 1 ciclo / 22 ns.

Figura 61. Simulación de tiempo del cifrador Hélix a 1 ciclo / 30ns.

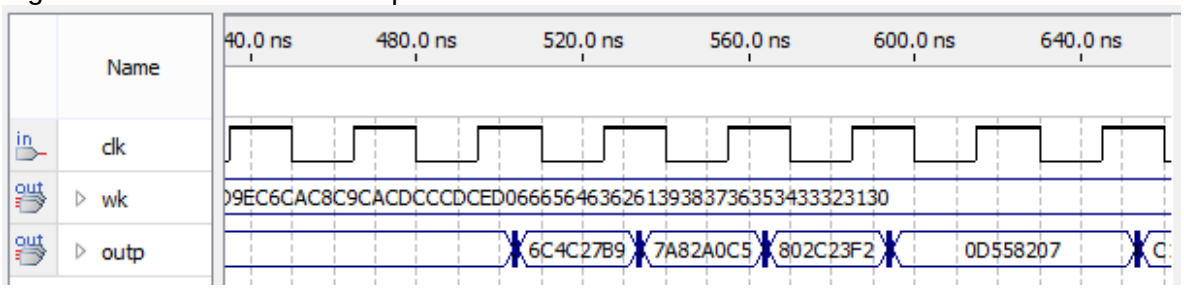
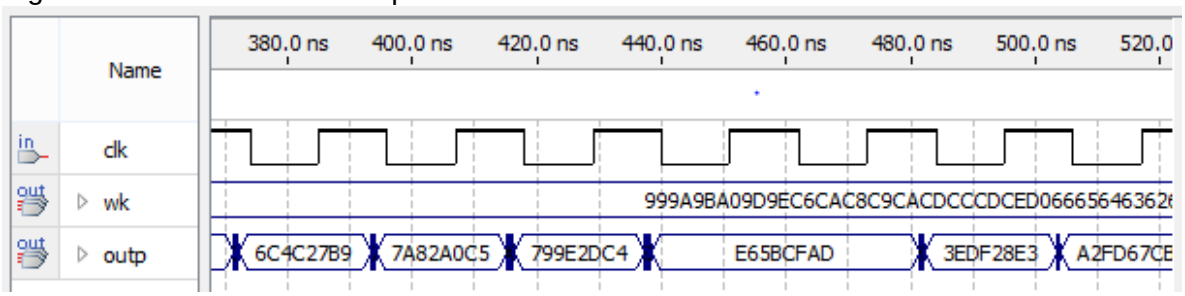


Figura 62. Simulación de tiempo del cifrador Hélix a 1 ciclo / 22ns



8.16 COMPARACIÓN ENTRE RESULTADOS

- Velocidad:

Tomando la velocidad mínima en que los cifradores operan correctamente según las simulaciones.

A5/1 = 10 ns por 1 bit

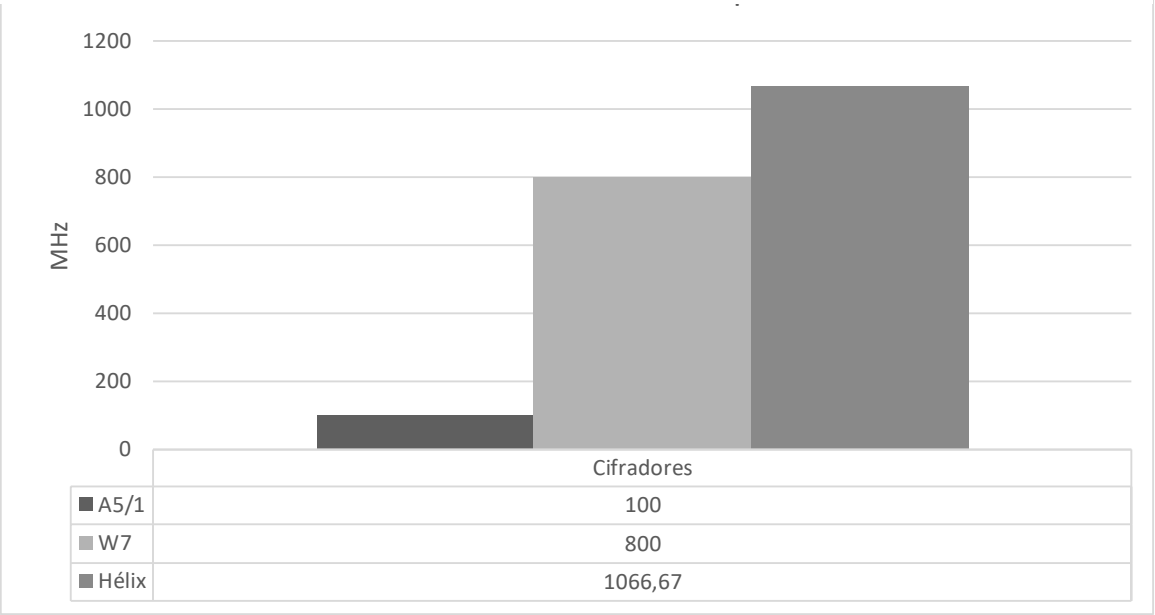
W7 = 10 ns por 8 bits

Hélix = 30 ns por 32 bits.

Los anteriores resultados consideran que los bits ingresan de forma paralela para el caso de W7 y Hélix, es decir que si los datos llegaran en secuencia de bits, se requeriría de un buffer que opere a una mayor velocidad para que envié la palabra de forma paralela coincidiendo con la velocidad a la que opera el cifrador. De ser así buffers de tal velocidad no son posibles en la FPGA, considerando que los circuitos en la FPGA especificada operan idóneamente a partir de 10ns.

Se especifica 10ns, ya que esta es la cifra promedio en que el simulador no reporta errores en un circuito sencillo dependiente de un reloj.

Figura 63. Velocidad de los cifradores por bit.



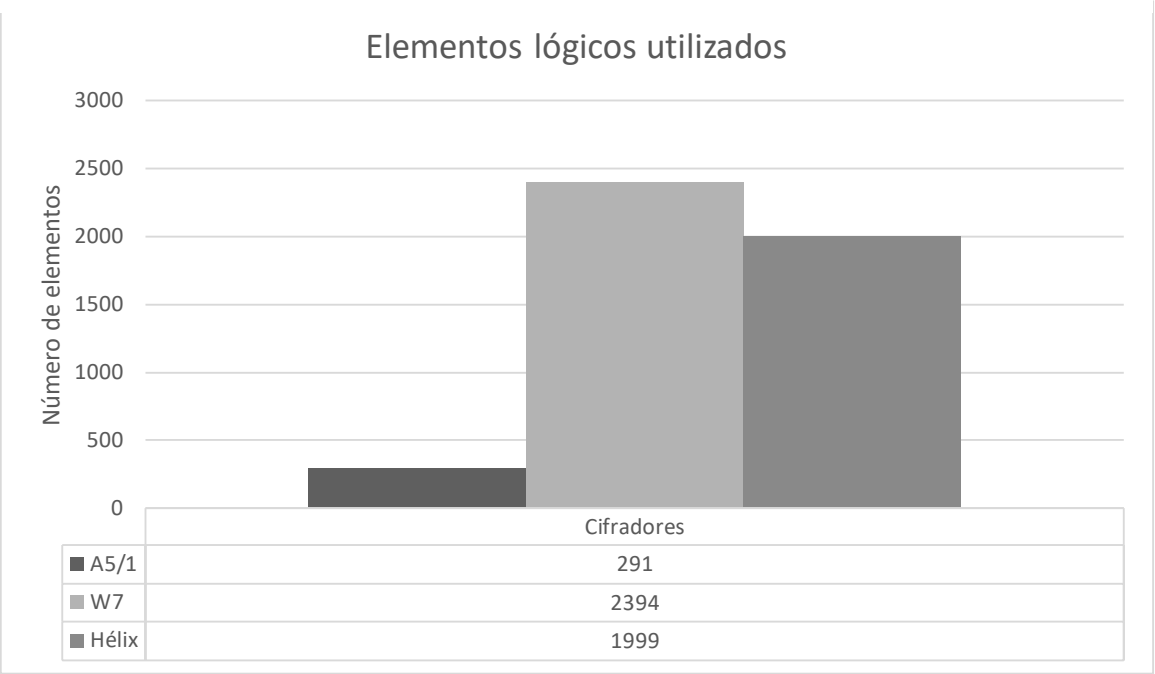
Conclusión:

Para aprovechar el máximo de la FPGA, se debe considerar su entrada de datos en paralelo, y no de forma serial, porque surge la necesidad de implementar buffers de mayor velocidad en el caso de una transmisión de datos serial.

- Recursos:

En la siguiente gráfica se resume el total de elementos lógicos utilizados por cada cifrador.

Figura 64. Elementos lógicos utilizados.



Conclusión:

El cifrador Hélix, resulta ser más eficiente en cuanto a W7, recordando el máximo número de bytes permitido por cada cifrador:

W7: 2^46 bytes (Especifica la secuencia máxima de bytes que se debe generar por clave).

Hélix: 2^64 bytes (Especifica longitud máxima de bytes que puede tener el texto).

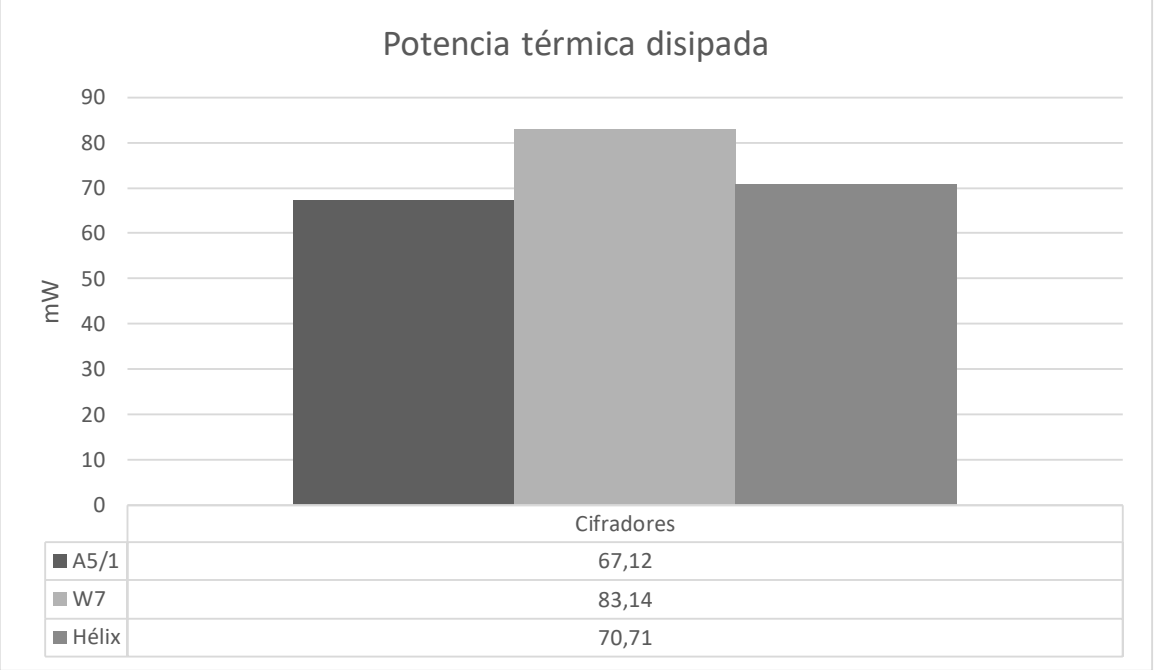
Hélix permite mayor longitud de texto y consume menos recursos lógicos.

El cifrador A5/1 para el desarrollo de un ASIC, resulta ser bastante económico por no requerir tantos recursos.

- Potencia:

Tomando potencia térmica total disipada

Figura 65. Potencia térmica disipada.



Conclusión:

Se esperaba que la potencia térmica disipada de A5/1 en cuando a los demás cifradores fuera mucho menor, por su menor uso de elementos lógicos. Sin embargo la diferencia si existe, pero no es linealmente proporcional al número de elementos lógicos utilizados.

Se concluye que mayor parte de la potencia disipada, se debe al funcionamiento general de la FPGA.

9. CONCLUSIONES

- Los resultados obtenidos del software Quartus II evidencia una aproximación del desempeño y costo de hardware usando como prototipo una FPGA. Que puede sugerir un sistema determinado según las condiciones requeridas. Por ejemplo, para un dispositivo móvil la prioridad sería tener un bajo costo de potencia eléctrica.

- Para aprovechar el máximo de la FPGA, se debe considerar su entrada de datos en paralelo, y no de forma serial, de lo contrario surge la necesidad de implementar buffers de mayor velocidad para los sistemas descritos.

- El cifrador Hélix, resulta ser más eficiente en cuanto a W7, recordando el máximo número de bytes permitido por cada cifrador

  W7: 2^46 bytes (Especifica la secuencia máxima de bytes que se debe generar por clave).

  Hélix: 2^64 bytes (Especifica longitud máxima de bytes que puede tener el texto).

  Hélix permite mayor longitud de texto y consume menos recursos lógicos.

- El cifrador A5/1 para el desarrollo de un ASIC, resulta ser bastante económico por no requerir tantos recursos. Al igual se esperaba que la potencia térmica disipada de A5/1 en cuando a los demás cifradores fuera mucho menor, por su menor uso de elementos lógicos. Sin embargo la diferencia si existe, pero no es linealmente proporcional al número de elementos lógicos utilizados, por lo que la mayor parte de la potencia disipada se debe al funcionamiento general de la FPGA.

- A5/1 y W7 son cifradores que se adaptan mejor a enlaces de telecomunicaciones, ya que los datos cifrados no tienen dependencia de otros, es decir, si se pierde uno, los demás siguen intactos, por eso se les conoce como cifradores de flujo síncronos.

- El Cifrador Hélix al ser dependiente del contenido del texto a cifrar y su longitud, no sugiere un uso apropiado en canales físicos de transmisión o en ambientes susceptibles, luego, la pérdida de un dato compromete al resto. Se recomienda usar este cifrador en software, considerando este un ambiente estable, además, si se quiere usar el mensaje de autenticación, requería almacenamiento extra en el descifrador para validar los datos con los mensajes (el que recibe y genera).

- La implementación de máquinas de estado para controlar cada proceso, permite ahorrar espacio y permite a la vez mayores velocidades al tener que usar todo bloque de construcción combinacional con un reloj común.

- En comparación a los cifradores W7 y Hélix, A5/1 tiene un tamaño de clave pequeño, pues a pesar de inicializar el cifrador con 86 bits, solo 64 bits son la clave privada y 22 bits representa el número de trama, por lo que un ataque de fuerza bruta requiere 2^64 bytes, cantidad reducida comparando con las tecnologías computacionales disponible hoy en día.

- El cifrador Hélix, tiene un bloque conformado de 20 rondas, que definiéndose de forma combinacional tiene un tiempo de respuesta mayor a los LFSR utilizados por A5/1 y W7.

- El cifrador W7 requiere una máquina de estados con menos etapas que A5/1 y es 8 veces más rápido que este al hacer uso de celdas que operan paralelamente, donde cada una cifra un bit. Posee una restricción de utilización por clave de 2^46 bytes, reduciendo la frecuencia de cambio de claves, que en A5/1 se realiza cada 228 bits.

- Los cifradores A5/1 y W7 son los candidatos más próximos a una implementación de hardware al hacer uso de operaciones XOR y desplazamiento de registros, que son ejercicios bastantes prácticos en hardware. Hélix en cambio añade suma aritmética, que en lógica booleana requiere un mayor uso de operaciones lógicas.

- Una de las ventajas importantes de los sistemas de cifrado implementados, es la no exigencia de hardware distinto para el descifrado de datos, excepto a Hélix que implementa el mensaje de autenticación (MAC), lo cual requiere memoria en el descifrador para almacenar el MAC que recibe, y compararla con la que este genera con el fin de validar los datos.

- El diseño de cifradores con los algoritmos propuestos garantiza privacidad de los datos y autenticidad para el caso del algoritmo Hélix. Sus respectivos diseños en VHDL permitió una fácil implementación en la FPGA gracias al software Quartus II Web Edition, que compila los diseños, realiza análisis temporales y configura el dispositivo (FPGA) desde el programador.

10. RECOMENDACIONES

Desde el punto de iniciación y finalización de este proyecto, surge un gran número de recomendaciones para aquellos interesados en el manejo de herramientas de diseño electrónico (EDA). Al igual recomendaciones de las posibles aplicaciones que tiene este proyecto.

Para aplicaciones:

- Cifradores para líneas de transmisión: Sin lugar a duda, gran parte de la información se maneja digitalmente, debido a todo a lo que aplica como transacciones financieras, comunicados de estado, contratos, comercio electrónico, etc., lo que requiere privacidad.

- Tarjetas de seguridad inteligentes: Estas tarjetas contienen un circuito integrado que ejecutan cierta lógica programada, no son tan usuales como las tarjetas magnéticas, pero son más robustas ante cambios electromagnéticos.

- Teclados para PIN: El caso más práctico de estos son los cajeros automáticos, permite al usuario identificarse mediante un número para acceder a ciertos sistemas.

- Cifrado generalizado: Debido a la independencia de las aplicaciones al cifrado de datos, ciertos sistemas de cómputo adoptan hardware embebido para excluir esta tarea del procesador anfitrión.

Para desarrollo usando herramientas EDA:

- Comprobación con herramientas: Hacer uso de herramientas como las hojas de cálculo de Excel permite comprobar junto con el simulador la correcta ejecución del circuito diseñado, de no coincidir, sugiere revisar el diseño, que por lo general son problemas de sincronización.

- Fragmentación del diseño: En el diseño de hardware lo idóneo es identificar lo que se puede definir de forma combinacional y hacer uso de estos mediante una máquina de estados, que define la lógica secuencial del diseño. Esto se refleja en el ahorro de recursos lógicos y la velocidad de desempeño.

- Considerar estados del reloj: Para mitigar y evitar problemas de sincronización y gastos de ciclos extra, no solo se debe considerar realizar todas las operaciones en flaco de subida o bajada.

- Definición de datos en VHDL: Definir datos en VHDL para su uso es menos engorroso que lidiar con los datos predefinidos de VHDL, por ejemplo, definir un tipo de dato para declarar vectores de enteros, facilita manipular posiciones de registros, memorias, etc.

- Construcción de funciones y componentes en VHDL: Las funciones son piezas de código para uso de propósito general como sumas, restas, divisiones, que en vez de definir repetitivamente en el código, de declaran una sola vez. Al igual los componentes son piezas de código reutilizables. Visualmente, el uso de estos hacen más entendible y manejable el código principal.

Para futuros trabajos se recomienda tomar algoritmos de cifrado de catálogos como eSTREAM, ECRYPT y de distintos proyectos relacionados a la criptografía, estos algoritmos son bastante competentes y vale la pena profundizar más en ellos para encontrar vulnerabilidades, posibles mejoras, y poder hacer implementaciones más eficientes.

Se sugiere realizar criptoanálisis asistido por hardware por medio de las FPGAs que son dispositivos rápidos y de capacidad considerable.

GLOSARIO

AGENTE: Persona o proceso que desea acceder a la información de un sistema (UNAM, 2012).

ALGORITMO: Conjunto de operaciones elementales que deben ser efectuadas para obtenerse un resultado deseado (Granados Paredes, 2006).

AMENAZA: Todo aquello que intenta o pretende violar la seguridad o causar daño a los recursos del sistema (UNAM, 2012).

AMENAZA EXTERNA: Todo aquello que intenta o pretende violar la seguridad o causar daño a los recursos del sistema (UNAM, 2012).

AMENAZA INTERNA: Empleados internos, socios y usuarios de confianza de una organización que intentan o pretenden violar la seguridad, estas personas están familiarizadas con la red, saben que sistemas contienen la información valiosa y pueden tener acceso a los sistemas a través de su propia cuenta o mediante la cuenta de otro usuario (UNAM, 2012).

ANÁLISIS DE RIESGO: Evaluación de amenazas y vulnerabilidades de la información y su impacto en el procesamiento de la información así como su probabilidad de ocurrencia (UNAM, 2012).

ATAQUE: Tentativa de criptoanálisis (González Rodríguez, 2012).

ATAQUE ACTIVO: Ataque que implica algún tipo de modificación del flujo de datos o la creación de un falso flujo de los mismos (UNAM, 2012).

ATAQUE PASIVO: Tipo de ataque que no altera en ningún momento la información, es decir, únicamente la observa, escucha, obtiene o monitorea mientras está siendo transmitida (UNAM, 2012).

CBC: Modo de cifrado. Del inglés "Cipher Block Chaining Mode" traduce "Cifrador de bloque de modo encadenado".

CFB: Modo de cifrado. Del inglés "Cipher Feedback Mode" traduce "Cifrador de modo retroalimentado.

CLAVES: Datos (llaves) privados/públicos que permiten cifrar un documento y descifrar el correspondiente criptograma (Mollin, 2007).

CIFRADO: Técnica protege o proporciona autenticidad a un documento o usuario al aplicar un algoritmo criptográfico. Sin conocer una clave específica o secreta, no será posible descifrarlo o recuperarlo (Moreno Eguílaz, 2012).

CONFIDENCIALIDAD: Propiedad de que la información no sea revelada ni puesta a disposición de las partes no autorizadas (UNAM, 2012).

CRIPTOANÁLISIS: La habilidad de descifrar criptogramas (Morales Sandoval, 2003).

CRIPTOANÁLISTA: Persona cuya función es romper algoritmos de cifrado en busca de debilidades, la clave o del texto en claro (Figueira, 2013).

CRIPTOGRAFÍA: Es la técnica de transformar un mensaje inteligible, denominado texto en claro, en otro que sólo puedan entender las personas autorizadas a ello, que llamaremos criptograma o texto cifrado (Granados Paredes, 2006).

CRIPTÓGRAFO: Máquina o artefacto para cifrar (Granados Paredes, 2006).

CRIPTOGRAMA: Mensajes cifrados (Granados Paredes, 2006).

CRIPTOLOGÍA: La síntesis de las técnicas de cifrado, conocido como criptografía, y aquellas técnicas de ataque conocidas como criptoanálisis (Granados Paredes, 2006).

CRIPTÓLOGO: Persona que trabaja de forma legítima para proteger la información creando algoritmos criptográficos (Granados Paredes, 2006).

CRIPTOSISTEMA / SISTEMA DE CIFRADO: Un sistema comprometido con algoritmos criptográficos, todo texto en claro, texto cifrado y claves (Network Associates, 1999).

CTR: Modo de cifrado. Del inglés "Counter Mode" traduce "Modo contador".

DESCIFRAR: Operación inversa de cifrar, o sea, obtener la versión original de un mensaje cifrado. Al contrario de la desencriptación, aquí se conoce el método descifrado. Proceso inverso que recupera el texto cifrado (González Rodríguez, 2012).

DIGRÁMICO: El mensaje en claro se cifra tomando sus caracteres por parejas, en lugar de carácter a carácter (UNAM, 2012).

ECB: Modo de cifrado. Del inglés " Electronic Code Book Mode " traduce "Modo libro de código electrónico."

ESTÁNDAR: Acuerdo documentado que contiene especificaciones técnicas u otros criterios precisos para ser utilizados, contiene reglas, guías o definiciones de características para asegurar que los materiales, productos, procesos y servicios son adecuados para su propósito (UNAM, 2012).

ESTEGANOGRAFÍA: Del griego "escritura escondida". Rama particular de la criptología que consiste, no en hacer con que un mensaje sea ininteligible, pero camuflado o enmascarando su presencia. Al contrario de la criptografía, que busca esconder la información del mensaje, la esteganografía busca esconder la existencia del mensaje (González Rodríguez, 2012).

FPGA: Dispositivo que permite implementar circuitos digitales (Pedroni, 2004).

HASH: Función matemática bien definida que tiene como valor de entrada cualquier tamaño y tipo de datos y como valor de salida una palabra de tamaño definido (Apokliptico, 2010).

INFORMACIÓN: Todo mensaje (conjunto de datos) que al receptor le interese, le entienda o lo ignore antes de recibirlo (UNAM, 2012).

INTEGRIDAD: Propiedad de asegurar que los datos se transmiten desde una fuente a un destino sin alteraciones no detectadas (UNAM, 2012).

INTRUSO: Aquel que puede realizar cualquier conjunto de acciones que puede comprometer la integridad, confidencialidad o disponibilidad de la información o un recurso informático (UNAM, 2012).

LACEDEMONIOS: Guerrero, relativo a la antigua Grecia (González Rodríguez, 2012).

MONOALFABÉTICO: Sustitución usando un único alfabeto. También llamada de sustitución simple (González Rodríguez, 2012).

OFB: Modo de cifrado. Del inglés "Output Feedback Mode" traduce "Modo de salida retroalimentado".

POLIALFABÉTICO: Un tipo de sustitución en la cual múltiples alfabetos de sustitución distinguidos son usados (González Rodríguez, 2012).

PROTOCOLO CRIPTOGRÁFICO: Serie de pasos registrada y autentificable para una red de computadoras o sistema distribuido (UNAM, 2012).

TEXTO PLANO / EN CLARO: Documento original, el texto que se debe proteger (Mollin, 2007).

TRANSPOSICIÓN: Reordenación de los elementos del mensaje (Lucena López, 2001).

SUSTITUCIÓN: Cambiar elementos; letras, códigos y símbolos, por otros del mensaje de acuerdo a un alfabeto (Lucena López, 2001).

BIBLIOGRAFÍA

Abd Al-Rasedy, A. S., & Al-Swidi, A. A. (Mayo de 2011). *An advantages and Dis Advantages of Block and Stream Cipher.* Obtenido de http://www.uobabylon.edu.iq/uobcoleges/fileshare/articles/block.pdf

Alfred J. Menezes, P. C. (1996). *Handbook of Applied Cryptography.* Obtenido de http://cacr.uwaterloo.ca/hac

Alfred-λ. (29 de Abril de 2005). *Apuntes de Criptografía III: Algoritmos Simétricos, Red de Feistel.* Obtenido de Zooglea: http://zooglea.blogspot.com/2005/04/apuntes-de-criptografa-iii-algoritmos.html

Altera. (2013). Obtenido de www.altera.com

Anderson, R. (2008). Cryptography. En R. Anderson, *Security Engineering* (págs. 129-184). Cambridge: Wiley. Obtenido de http://www.cl.cam.ac.uk/~rja14/Papers/SE-05.pdf

Apokliptico. (21 de 11 de 2010). *El Hacker*. Recuperado el 30 de 10 de 2012, de Glosario de términos de Criptografía: http://foro.elhacker.net/criptografia/glosario_de_teminos_de_criptografia_elhackernet-t311330.0.html

Arévalo Aldana, J. A., González, R. A., & Sánchez, J. C. (2009). Fundamentos y principios de criptografía. *Conferencia (Diapositivas)*. Bogotá, Colombia.

Bustos Pérez, J. Á. (2002). Criptografía . *Conferencia (Diapositivas)*. Obtenido de http://es.tldp.org/Presentaciones/200203jornadassalamanca/jadebustos/conferencia-criptografia.pdf

Cachin, C. (Dirección). (2012). *Storage encryption and key management* [Película]. United Kingdom.

Chandler, D. L. (15 de Febrero de 2011). *Rivest Unlocks Cryptography's Past, Looks toward Future.* Obtenido de MIT news: http://web.mit.edu/newsoffice/2011/killian-rivest-0215.html

Coffey, N. (18 de Marzo de 2013). *Comparison of ciphers*. Obtenido de JavaMex: http://www.javamex.com/tutorials/cryptography/ciphers.shtml

Díaz, E. M., & Arévalo Aldana, J. A. (2009). *Encryption and Decryption FEAL Algorithm in VHDL.* Bogotá: Fundación Universitaria San Martín.

Diffie, W., & Hellman, M. E. (1976). *New Directions in Criptography.*

Douglen, A. (Noviembre de 2010). *security stactk exchange*. Obtenido de Advantages and disadvantages of Stream versus Block Ciphers: http://security.stackexchange.com/questions/334/advantages-and-disadvantages-of-stream-versus-block-ciphers

ECRYPT. (2013). *Notes on the ECRYPT Stream Cipher project (eSTREAM).* Obtenido de http://cr.yp.to/streamciphers.html

Ferguson, N., Whiting, D., Schneier, B., Kelsey, J., Lucks, S., & Tadayoshi, K. (2003). *Helix Fast Encryption and Authentication in a Single Cryptographic Primitive.*

Figueira, C. (21 de Febrero de 2013). *Criptografía y seguridad de datos.* Recuperado el 29 de Agosto de 2013, de Universidad Simón Bolivar: http://ldc.usb.ve/~figueira/cursos/Seguridad/Material/Cortafuegos.pdf

Goldwasser, S., & Bellare, M. (2008). Introduction to Modern Cryptography. En S. Goldwasser, & M. Bellare, *Lecture Notes on Cryptography* (págs. 11-15). Obtenido de http://cseweb.ucsd.edu/~mihir/papers/gb.pdf

González Rodríguez, M. (2012). *Glosario.* Obtenido de Facultad de Informática de la Universidad de Las Palmas de Gran Canaria: http://serdis.dis.ulpgc.es/~ii-cript/PAGINA%20WEB%20CLASICA/GLOSARIO.html

Granados Paredes, G. (10 de Julio de 2006). Introducción a la criptografía. *DGSCA-UNAM Revista Digital Universitaria, VII*(7), 1-17. Obtenido de http://www.revista.unam.mx/vol.7/num7/art55/jul_art55.pdf

Hietala, J. (Septiembre de 2007). *Hardware versus Software.* Obtenido de Seagate: http://www.seagate.com/staticfiles/SeagateCryptofaceoff.pdf

Hiremath, S., & Suma, M. S. (2009). Advance Encryption Standard Implemented on FPGA. *Second International Conference on Computer and Electrical Engineering*, 656-660.

Ibrahimy, M. I., Reaz, M. B., Asaduzzaman, K., & Hussain, S. (2007). FPGA Implementation of RSA Encryption Engine with Flexible Key Size. *International Journal of Communications*, 107-113.

ITESCAM. (Mayo de 2004). *Historia de la criptografía.* Obtenido de ITESCAM: http://www.itescam.edu.mx/principal/sylabus/fpdb/recursos/r68645.PDF

Kingston. (17 de Marzo de 2013). *Hardware- vs Software-Based Encryption*. Obtenido de Kingston Technology: http://www.kingston.com/us/usb/encrypted_security/hardware_vs_software

Kroll. (2013). *Global Faud Report.* Obtenido de Kroll Advisory Solutions: http://www.krolladvisory.com/insights-reports/global-fraud-reports/

Kruh, L., & Deavours, C. (Enero de 2002). *The Commercial Enigma: Beginnings of Machine Cryptography.* Obtenido de http://www.nymphomath.ch/crypto/bibliotheque/PDF/KruhDeavours.pdf

Lerch, D. (22 de Julio de 2007). *Modos de cifrado: ECB, CBC, CTR, OFB y CFB*. Recuperado el 29 de Agosto de 2013, de El Blog de Daniel Lerch: http://dlerch.blogspot.com/2007/07/modos-de-cifrado-ecb-cbc-ctr-ofb-y-cfb.html

Lucena López, M. J. (Junio de 2001). *Criptografía y Seguridad en Computadores.* Obtenido de IIE de la Universidad de la República: http://iie.fing.edu.uy/ense/asign/seguro/Criptografia.pdf

Ma Jianfeng, Z. Y., & Dong Lihua, P. Q. (2009). A Hardware-Oriented Fast Enryption Keystream for Digital Rights Management. *International Conference on Computational Intelligence and Security*, 584-586.

Macé, F., Standaert, F. X., & Quisquater, J. J. (Febrero de 2008). FPGA Implementation(s) of a Scalable Encryption Algorithm. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems, 16*(2), 212-216.

Mancilla Tello, J. E. (1995). *Implementación de los nuevos avances en criptografía: Un aporte al desarrollo en la transferencia electrónica de datos en Guatemala.* Univesidad Francisco Marroquín, Ingeniería de Sistemas.

Mollin, R. A. (2007). *An Introduction to Cryptography* (Segunda ed.). Boca Ratón, Flórida, Estados Unidos: Chapman & Hall/CRC.

Morales Sandoval, M. (2003). *Notas sobre criptografía.* Obtenido de Cinvestav Tamaulipas: http://www.tamps.cinvestav.mx/~mmorales/documents/Criptograf.pdf

Moreno Eguílaz, M. (2012). Criptografía. Obtenido de Universitat Politècnica de Catalunya: http://tec.upc.es/sda/Fundamentos%20Criptografia.pdf

National Security Agency. (2008). *Navajo Code Talkers.* Obtenido de NSA: http://www.nsa.gov/about/_files/cryptologic_heritage/publications/wwii/navajo_codetalkers.pdf

Network Associates, I. (1999). *An Introduction to Cryptography.* Santa Clara, California, Estados Unidos. Obtenido de ftp://ftp.pgpi.org/pub/pgp/6.5/docs/english/IntroToCrypto.pdf

Pardo Carpio, F. (11 de Octubre de 1997). *VHDL Lenguaje de descripción y modelado de circuitos.* Recuperado el 29 de Agosto de 2013, de Universidad Nacional de Rosario: http://www.dsi.fceia.unr.edu.ar/downloads/DDA/vhdl_PardoCarpio.pdf

Pedroni, V. A. (2004). Circuit design with VHDL. En V. A. Pedroni, *Circuit design with VHDL* (págs. 4-5). TLFeBOOK.

Protechnix. (2013). *Cryptology and Data Secrecy : The Vernam Cipher.* Obtenido de Pro Technix: http://www.pro-technix.com/information/crypto/pages/vernam_base.html

Saluja, K. K. (1991). *Linear Shift Registers Theory and Applications.*

Sánchez Martínez, M. (Julio de 2005). *Diseño en FPGA de un Circuito Comparador de Imágenes.* Recuperado el 29 de Agosto de 2013, de CINVESTAV: http://www.cs.cinvestav.mx/TesisGraduados/2005/tesisMiguelAngelS.pdf

Schmidt, T. (18 de Marzo de 2013). *The History of Cryptography.* Obtenido de UNC Charlotte: http://coedpages.uncc.edu/cstem/summer%20ventures/2010%20History%20of%20Math/Taylor.pdf

Schneier, B. (1999). *A Self-Study Course in Block-Cipher Cryptanalysis.* Obtenido de Schneier: http://www.schneier.com/paper-self-study.pdf

Soto Campos, J. G. (15 de Septiembre de 2004). *Estadísticas sobre delitos informáticos.* Obtenido de emagister: http://www.emagister.com/curso-delitos-informaticos/estadisticas-sobre-delitos-informaticos

Thomas, S., & Anthony, D. (Octubre de 2002). *The W7 Stream Cipher Algorithm.*

UNAM. (2012). *Fundamentos de criptografía.* Obtenido de UNAM - Facultad de Ingeniería: http://redyseguridad.fi-p.unam.mx/proyectos/criptografia/criptografia/index.php/glosario?showall=&start=1

UNM, E. (18 de Marzo de 2013). *Hardware Design with VHDL.* Obtenido de The University of New Mexico: http://www.ece.unm.edu/~jimp/vhdl_fpgas/slides/pld1.pdf

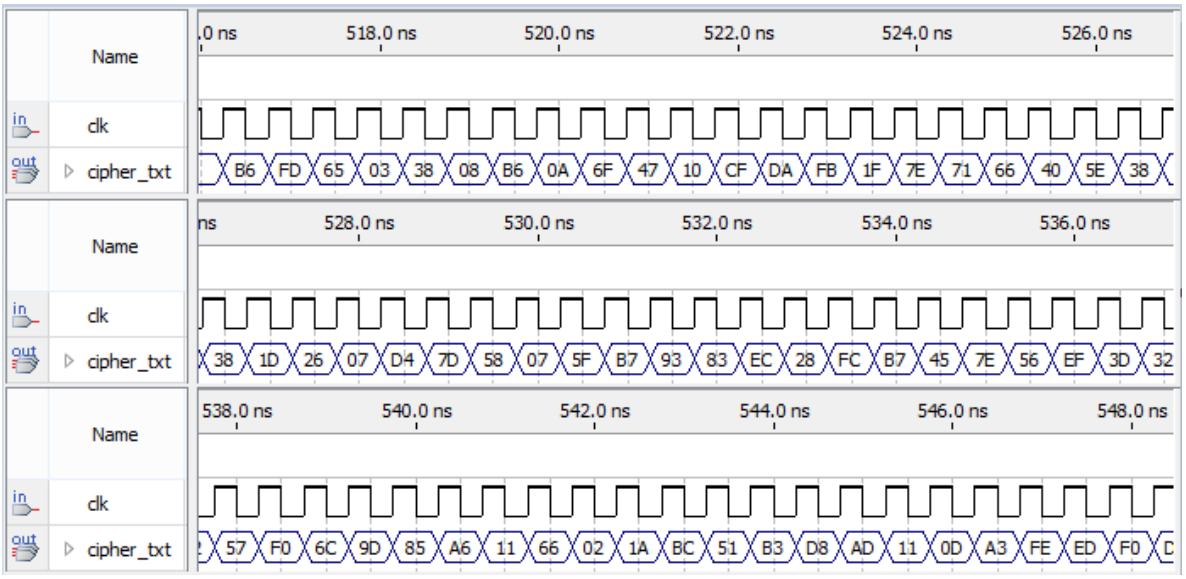
Vargas, J. (19 de Junio de 2006). *Álgebra Clásica*. Obtenido de Sociedad Matemática Mexicana: http://www.smm.org.mx/SMMP/html/modules/Publicaciones/PE/text7/AlgClas-orig.pdf

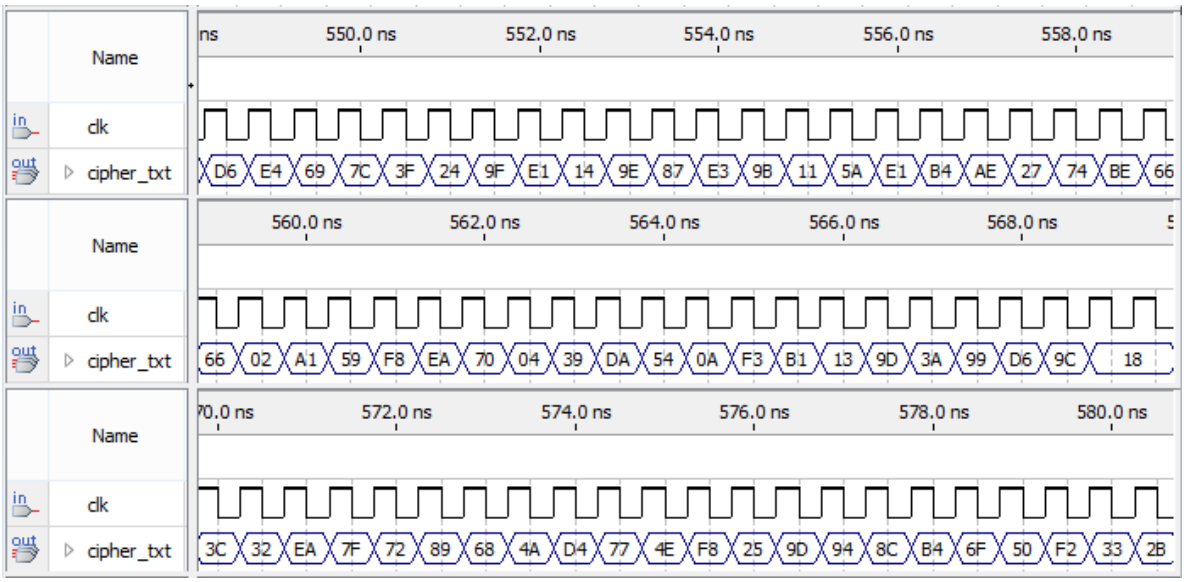
VIA. (16 de Marzo de 2013). *Why hardware encryption is better than software encryption*. Obtenido de VIA Technologies Inc: http://www.via.com.tw/en/initiatives/padlock/whyhardwareisbetter.jsp

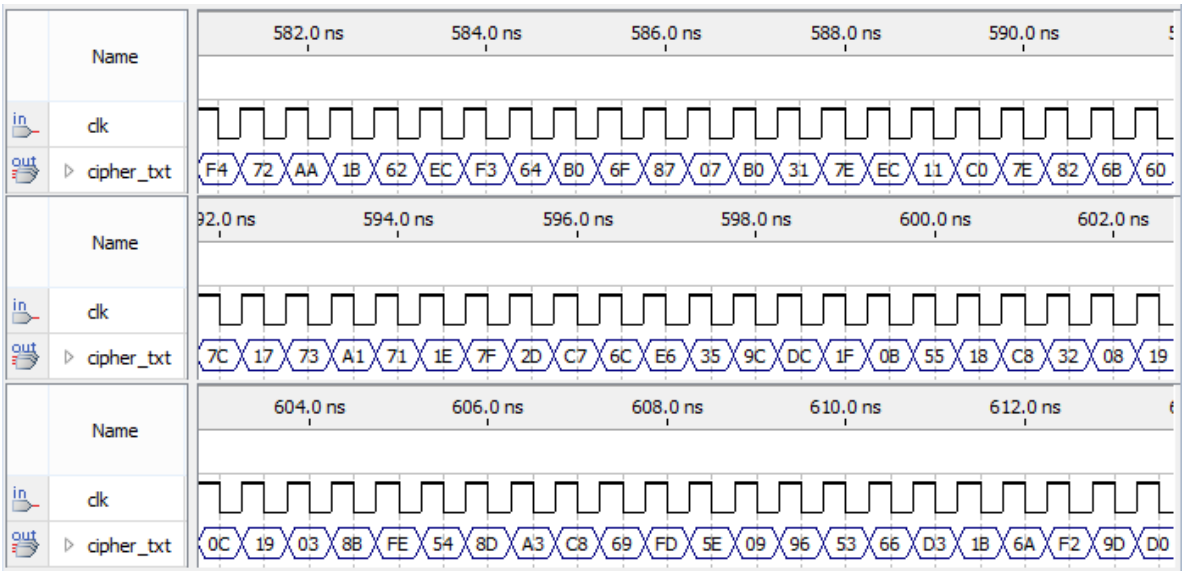
Xianwei, G., Erhong, L., Li, L., & Lang, K. (2008). LUT-based FPGA Implementation of SMS4/AES/Camellia. *Fifth IEEE international Symposium on Embedded Computing*, 73-76.

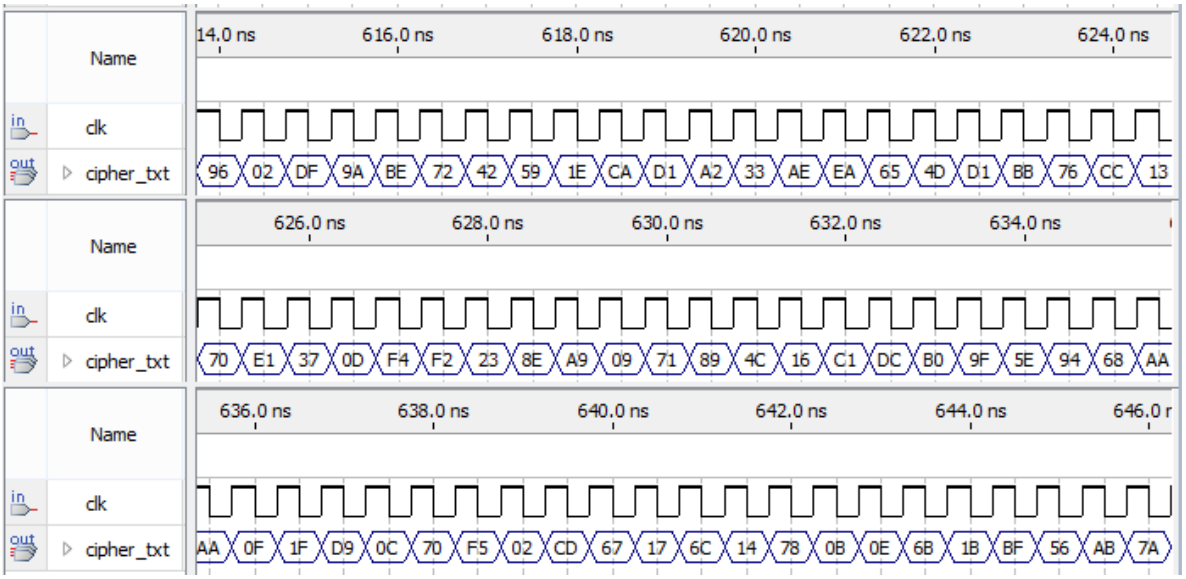
ANEXOS

Anexo A. Texto cifrado por el cifrador W7.






Name
582.0 ns
584.0 ns
586.0 ns
588.0 ns
590.0 ns
clk
cipher_txt
F4 72 AA 1B 62 EC F3 64 B0 6F 87 07 B0 31 7E EC 11 C0 7E 82 6B 60
Name
92.0 ns
594.0 ns
596.0 ns
598.0 ns
600.0 ns
602.0 ns
clk
cipher_txt
7C 17 73 A1 71 1E 7F 2D C7 6C E6 35 9C DC 1F 0B 55 18 C8 32 08 19
Name
604.0 ns
606.0 ns
608.0 ns
610.0 ns
612.0 ns
clk
cipher_txt
0C 19 03 8B FE 54 8D A3 C8 69 FD 5E 09 96 53 66 D3 1B 6A F2 9D D0


Name
14.0 ns
616.0 ns
618.0 ns
620.0 ns
622.0 ns
624.0 ns
clk
cipher_txt
96 02 DF 9A BE 72 42 59 1E CA D1 A2 33 AE EA 65 4D D1 BB 76 CC 13
Name
626.0 ns
628.0 ns
630.0 ns
632.0 ns
634.0 ns
clk
cipher_txt
70 E1 37 0D F4 F2 23 8E A9 09 71 89 4C 16 C1 DC B0 9F 5E 94 68 AA
Name
636.0 ns
638.0 ns
640.0 ns
642.0 ns
644.0 ns
clk
cipher_txt
AA 0F 1F D9 0C 70 F5 02 CD 67 17 6C 14 78 0B 0E 6B 1B BF 56 AB 7A

Análisis Comparativo de los Cifradores A5/1, W7 y Hélix (Noviembre 2013)

D. Y. Cañón. *Miembro Estudiantil, IEEE*

***Abstract*— As securing data has always been an issue, the number of cipher algorithms has considerably grown, offering a wide range to choose from. Like services tends to be mobile, the wireless networks get extra attention. The GSM networks are secured by the A5/1 algorithm, and the W7 algorithm has been proposed as a formal for its replacement for higher security. The Helix has no formal used in telecommunications, but its authors suggest to be strong enough and even faster than AES. In this paper these algorithms are implemented in VHDL, using the Altera FPGA EP2C20F484C7N as target device and are compared using the results from Quartus II simulator and Quartus II compiler summary.**

***Index Terms*— A5/1, FPGA, Helix, LFSR, Stream Cipher, VHDL, W7.**

I. Introducción

Las redes GSM utiliza el algoritmo A5/1 para cifrar los datos de voz, el cifrador genera una secuencia pseudo-aleatoria basado en LFSRs que opera los datos sin formato mediante una OR exclusiva, estos datos se transmiten en tramas de 228 bits. El estado inicial del cifrador depende de una clave privada de 64 bits que se genera en el establecimiento de una conversación y del número de trama de 22 bits. El algoritmo W7 es análogo a A5/1 y ha sido propuesto como su reemplazo por cuestiones de seguridad, a diferencia de A5/1 el cifrado se hace por secuencia de bytes al hacer uso paralelo de celdas análogas $C_7, C_6, ..., C_0$, el estado inicial del cifrador depende de una clave simétrica de 128 bits. El algoritmo Hélix cifra los datos por secuencia de palabras dobles, su estado inicial depende de una clave privada de longitud variable, y del nonce. Este cifrador basa su aleatoriedad en un bloque constituido de 20 rondas, donde cada ronda corresponde a la rotación de una palabra doble un número determinado de veces y la adición de esta palabra al estado de otra palabra, estas rondas se aplican de forma helicoidal atreves de los 5 estados de la palabra que conforman el bloque [1].

Se realizó la implementación de cada uno de los algoritmos de cifrado en VHDL y como dispositivo objetivo se usó la FPGA Altera EP2C20F484C7N, se provee una comparación de los cifradores en cuanto a elementos lógicos utilizados, potencia térmica disipada y velocidad mínima de ejecución al tomar los resultados del compilador y simulador de Quartus II.

D. Y. Cañón estudia en la Fundación Universitaria San Martín, Cra. 19 No 80-63, Bogotá D.C. – Colombia (e-mail: dc082040@ingenieria.sanmartin.edu.co)

II. Algoritmos de cifrado

A. *Cifrador A5/1*

A5/1 [1] se compone de tres LFSRs de 19, 22 y 23 bits, y en el mismo orden son retroalimentados por la operación XOR de los polinomios x^19 + x^18 + x^17 + x^14 + 1, x^22 + x^21 + 1 y x^23 + x^22 + x^21 + x^8 + 1. Cada LFSR se desplaza utilizando ciclos de reloj que son determinados por la función mayoritaria, la cual toma un bit de cada registro, el cual se denomina como bit habilitador, la función tomará el valor del que sea mayoría, 0 o 1, y de la misma manera si el bit habilitador coincide con la función, se desplazará el LFSR.

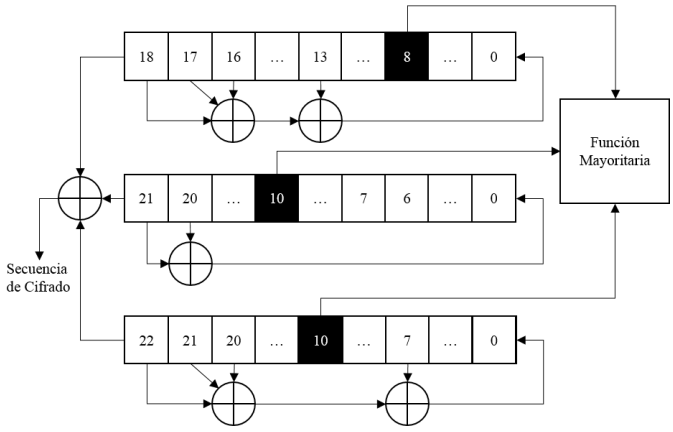


Fig. 1. Arquitectura de A5/1.

Para generar la secuencia de cifrado se requiere de 5 pasos. El primer paso es inicializar los LFSRs con cero. El segundo paso ignorando la función mayoritaria, se desplaza los LFSRs alimentándolos con la operación XOR entre el bit de clave y el polinomio de retroalimentación, esto se realiza 64 veces, lo cual corresponde a la longitud de la clave privada. El tercer paso es igual al segundo, con la excepción de que se desplaza 22 veces los registros con cada bit del número de trama. El cuarto paso desplaza los LFSRs 100 veces teniendo en cuenta la función mayoritaria. Y el último paso opera el texto plano con la secuencia de cifrado mediante una XOR, se hace 228 veces y corresponde a la longitud de la trama.

B. *Cifrador W7*

El algoritmo W7 [2] se compone de 8 celdas, donde cada una está compuesta de una función mayoritaria y 3 LFSRs. Cada celda tiene distintas configuraciones por función de retroalimentación y de salida filtrada de cada LFSR. Básicamente, la función de retroalimentación es una operación XOR entre determinadas posiciones del LFSR y la salida filtrada es la operación XOR entre conjuntos de posiciones del LFSR operados con AND.

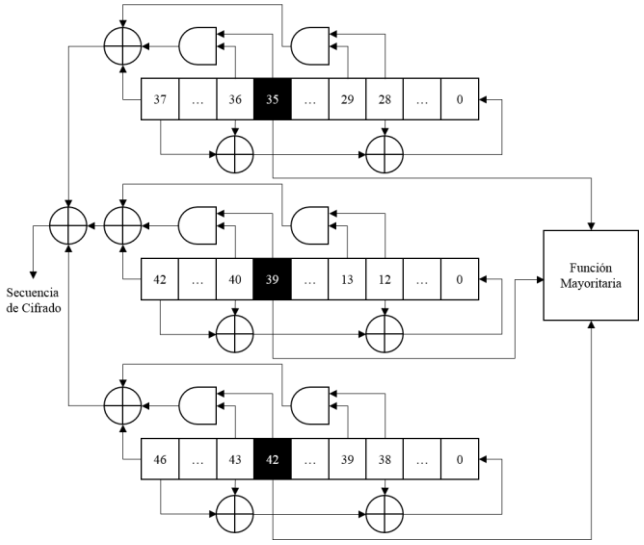


Fig. 2. Ejemplo de una celda de W7.

Los LFSRs para todas las celdas son de 38, 43 y 47 bits de longitud.

El cifrado se establece en tres pasos. El primer paso inicializa todos los LFSRs con la clave privada, donde el MSB corresponde al MSB del LFSR de 47 bits y el LSB al LSB del LFSR de 38 bits. El segundo paso desplaza los LFSRs 1031 veces. Y el tercer paso cifra el texto plano por bytes.

C. *Hélix*

El algoritmo Hélix [3] cifra por palabras dobles, la estructura fundamental del algoritmo es la ronda, que consiste de cinco estados de palabra, una adición XOR o aritmética y una rotación (ROL). La ronda consiste en la adición o XOR de una palabra a la otra, y la rotación de la primera palabra un número determinado veces, en la figura 3 se ilustra la ronda.

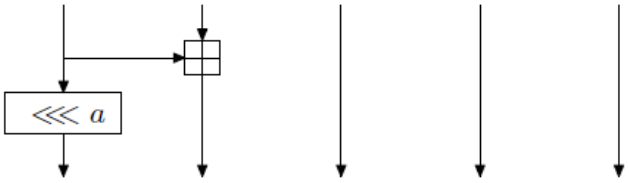


Fig. 3. Una Ronda de Hélix [3].

Pero la secuencia de cifrado se genera a partir de la construcción del bloque Hélix, el cual está conformado de 20 rondas con tres entradas (2 palabras claves y el texto plano) y una salida (secuencia de cifrado), en la figura 4 se ilustra el bloque hélix. Un bloque hélix corresponde al cifrado de una palabra doble, por lo tanto será necesario tantos bloques como palabras dobles hayan por cifrar, donde la salida de cada bloque alimentará la entrada del próximo.

El cifrador depende de una clave variable de 0 - 256 bits y un nonce de 128 bits, el cual es extendido por la ecuación 1.

$$N_k := (k \bmod 4) - N_{k-4} (\bmod 2^{32}) \quad (1)$$
$$k = 4,...,7.$$

El bloque Hélix es utilizado para crear una función de ronda F, que mapea 128 bits a 128 bits, donde las 4 entradas a F son extendidas con una palabra con valor $l(u)+64$ para completar las 5 palabras requeridas por el bloque Hélix, sus demás entradas tienen valor cero. Esta función se utiliza para generar la clave trabajo partiendo de la clave privada, la cual se extiende con $32 - l(U)$ bytes y se convierte a 8 palabras $K_{32},..., K_{39}$, En la ecuación 3 se define las palabras restantes, donde $K_0,..., K_7$ conforman la clave de trabajo.

Con la clave privada se genera la clave de trabajo de 256 bits, con esta clave el cifrador genera las palabras claves para los bloques de Hélix, las cuales se definen en la ecuación 2.

$$X_{i,0} := K_{i,0} \bmod 8$$
$$X_{i,1} := K_{(i+4) \bmod 8} + N_{i \bmod 8} + X'_i + i + 8 \quad (2)$$
$$X'_i := \begin{cases} \lfloor (i+8)/2^{31} \rfloor \Rightarrow i \bmod 4 = 3 \\ 4 \cdot l(u) \Rightarrow i \bmod 4 = 1 \\ 0 \end{cases}$$

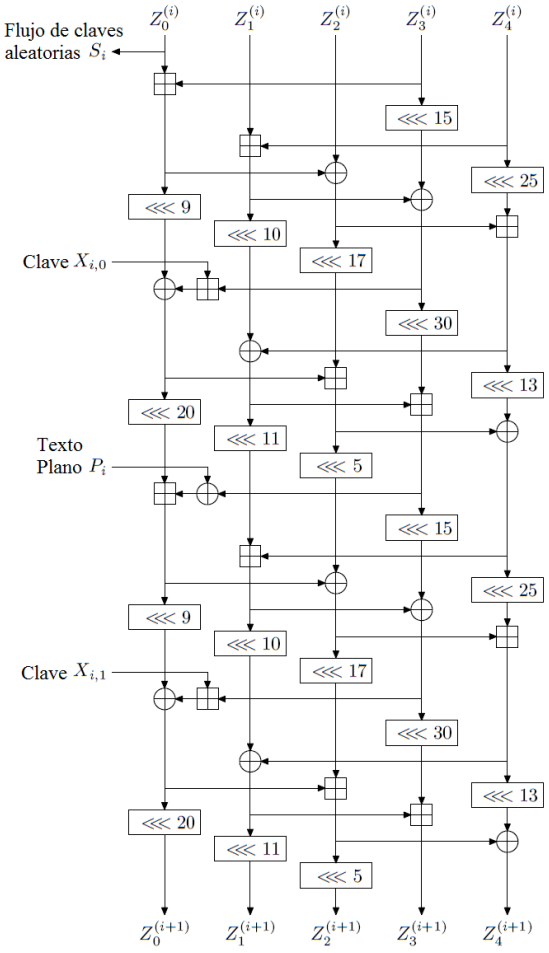


Fig. 4. Un bloque de Hélix [3].

$$(K_{4i},...,K_{4i+3}) := F((K_{4i+4},...,K_{4i+7})) \oplus (K_{4i+8},...,K_{4i+11}) \quad (3)$$

$i = 7,...,0$

El estado inicial del cifrador se define por la ecuación 4, donde el texto plano es cero y la secuencia de cifrado se descarta. Ocho bloques se aplican de -8 a -1.

$$Z_i^{(-8)} = K_{i+3} \oplus N_i \Rightarrow i = 0,...,3 \quad (4)$$

$$Z_4^{(-8)} = K_7$$

Concluida la iniciación se procede a cifrar el texto plano al aplicar un número *K* de bloques, donde $K := \lfloor (l(P)+3)/4 \rfloor$.

III. VHDLY FPGA

Muchas empresas que crean productos de diseño como FPGAs, CPLDs y procesadores enfocan sus sistemas a hacer implementaciones por lenguaje de descripción de hardware, que entre los más populares está VHDL permitiendo generar gran variedad de implementaciones.

A. VHDL

VHDL es un lenguaje de descripción de hardware en donde se especifica las características lógicas requeridas para un comportamiento digital esperado. VHDL es un sinónimo de VHSIC que es la abreviación de circuitos integrados de alta capacidad (Very High Speed Integrated Circuits) iniciado en el departamento de defensa de Estados Unidos de América en la década de los 80 [4].VHDL a lo largo de los años ha ido evolucionando desde su primer versión VHDL 87, el VHDL 93 y por último el VHDL 2008 estandarizado como un lenguaje de descripción de hardware IEEE 1076 pero tiene estándares adicionales como IEEE 1164 para introducir sistema lógico de valores múltiples.

El proceso de crear un código VHDL inicia con la creación de una entidad que tiene como extensión .vhd y el nombre de la entidad anticipándolo. El primer paso en el proceso de síntesis es la compilación. Compilación es la conversión del lenguaje VHDL de alto nivel, donde se describe el circuito en el nivel de transferencia de registros (RTL), en una lista de conexiones en el nivel de compuertas. El segundo paso es la optimización, que se realiza en la lista de conexiones a nivel de compuertas para la velocidad o para el área. En esta etapa, el diseño puede ser simulado. Por último, un software de localización y enrutamiento (instalador) va a generar la disposición física para un chip PLD / FPGA o va a generar las máscaras para un ASIC [4].

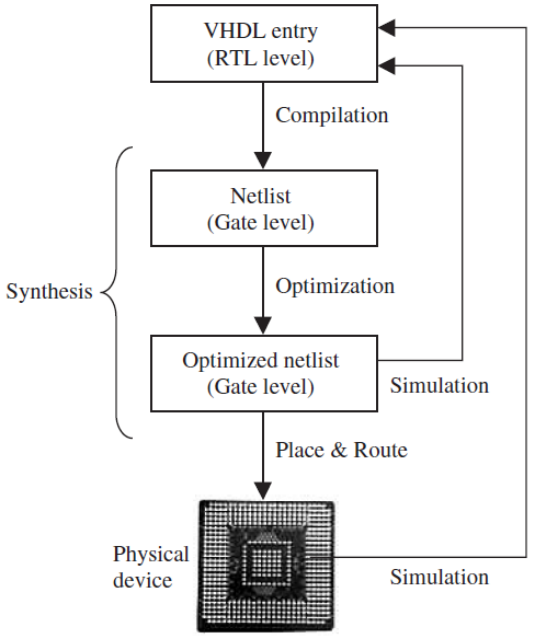


Fig. 5. Diseño de flujo VHDL [4].

B. FPGA de Altera

ALTERA es un fabricante de instrumentación lógica fundada en 1983 en San José, California la cual es pionera en la lógica programable basada en algoritmos de sintonización por medio de microchips programables con memoria embebida y conmutadores lógicos inteligentes. Hoy en día cuenta con un portafolio de diseño general que proporciona versatilidad en cualquier campo de la electrónica. ALTERA divide sus dispositivos en familias relacionadas con diseño de fabricación tanto en las FPGAs, ASICs, CPLDs y otros dispositivos de categoría específica (por ejemplo equipos militares), en software tiene una gama de programas robustos en la configuración y simulación de circuitos digitales de complejidad como Quartus II y Nios II (para la programación de procesadores embebidos), entre otros [5].

Los dispositivos Cyclone II tiene procesadores NIOS II que permiten soluciones integradas de ajuste personalizado. Los dispositivos Cyclone II poseen también gran variedad de periféricos, gran capacidad de memoria, gran variedad de entradas y salidas y un óptimo rendimiento de procesadores embebidos como lo son los NIOS II de tipo simple o múltiple para ser vinculados con otros procesadores o reemplazar en algunos casos. La unión de Cyclone II y Nios II permite soluciones de procesamiento integrado de alto desempeño y gran complejidad a bajo costo. Además de los componentes, los dispositivos Cyclone II permiten una amplia gama de interfaces y protocolos de entrada y salida utilizando la función Fast-on que ofrece un tiempo más rápido de entendimiento entre dispositivos [5].

Fig. 6. FPGA EP2C20F484C7N.

IV. RESULTADOS

Utilizando las pruebas hechas en el momento que se implementaron los 3 sistemas de cifrado, se pudo hacer una comparación que relaciona los recursos utilizados en la FPGA EP2C20F484C7:

A. Velocidad.

Tomando la velocidad mínima en que los cifradores operan correctamente según las simulaciones.

- A5/1: 10 ns por 1 bit.
- W7: 10 ns por 8 bits.
- Hélix: 30 ns por 32 bits.

Los anteriores resultados consideran que los bits ingresan de forma paralela para el caso de W7 y Hélix, es decir que si los datos llegaran en secuencia de bits, se requeriría de un buffer que opere a una mayor velocidad para que envié la palabra de forma paralela coincidiendo con la velocidad a la que opera el cifrador. De ser así buffers de tal velocidad no son posibles en la FPGA, considerando que los circuitos en la FPGA especificada operan idóneamente a partir de 10ns.

Se especifica 10ns, ya que esta es la cifra promedio en que el simulador no reporta errores en un circuito sencillo dependiente de un reloj.

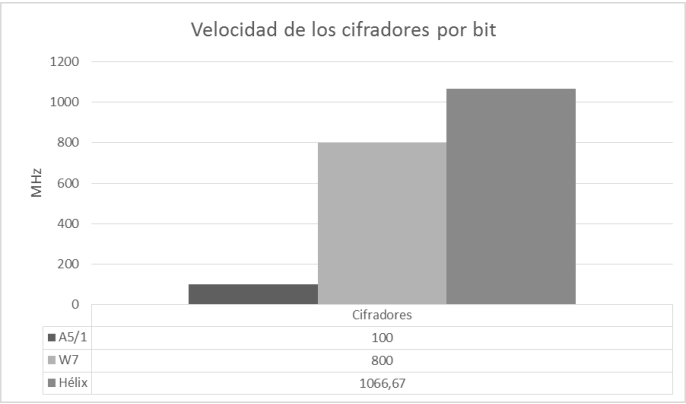


Fig. 7. Velocidad de cifrado por bit.

Para aprovechar el máximo de la FPGA, se debe considerar su entrada de datos en paralelo, y no de forma serial, porque surge la necesidad de implementar buffers de mayor velocidad en el caso de una transmisión de datos serial.

B. Recursos.

Teniendo en cuenta los elementos lógicos que se necesitaron en la implementación de cada uno de los cifradores en la FPGA mencionada, En la siguiente gráfica se resume el total de elementos lógicos utilizados por cada cifrador.

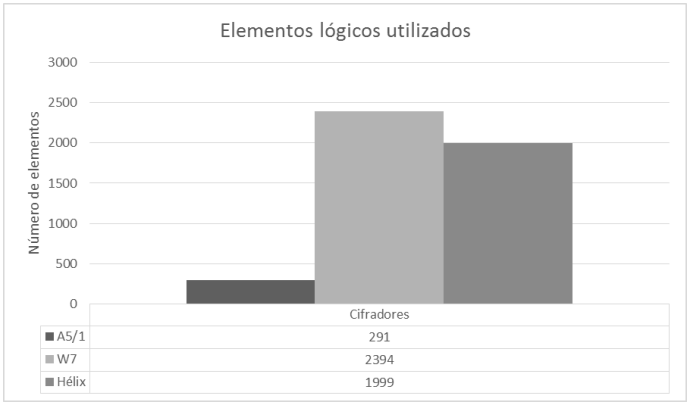


Fig. 8. Elementos lógicos utilizados.

El cifrador Hélix, resulta ser más eficiente en cuanto a W7, recordando el máximo número de bytes permitido por cada cifrador:

W7: 2^46 bytes (Especifica la secuencia máxima de bytes que se debe generar por clave).

Hélix: 2^64 bytes (Especifica longitud máxima de bytes que puede tener el texto).

Hélix permite mayor longitud de texto y consume menos recursos lógicos.

El cifrador A5/1 para el desarrollo de un ASIC, resulta ser bastante económico por no requerir tantos recursos.

C. Potencia.

La potencia térmica disipada es obtenida por medio de analizador que se adquiere en el momento que se compila el programa y es implementado en la FPGA.

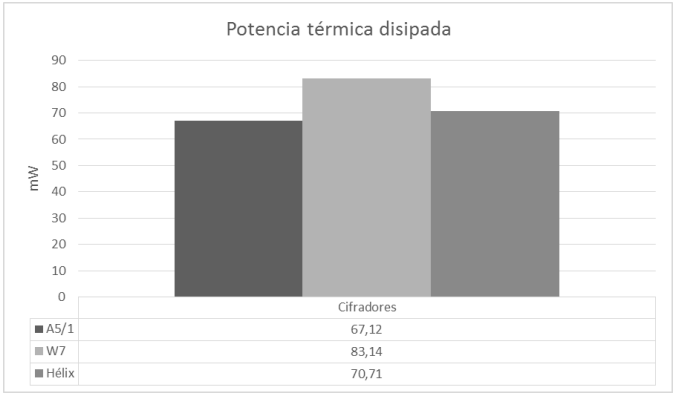


Fig. 9. Potencia termina disipada.

Se esperaba que la potencia térmica disipada de A5/1 en cuando a los demás cifradores fuera mucho menor, por su menor uso de elementos lógicos. Sin embargo la diferencia si

existe, pero no es linealmente proporcional al número de elementos lógicos utilizados.

Se concluye que mayor parte de la potencia disipada, se debe al funcionamiento general de la FPGA.

V. Conclusión

A5/1 y W7 son cifradores que se adaptan mejor a enlaces de telecomunicaciones, ya que los datos cifrados no tienen dependencia de otros, es decir, si se pierde uno, los demás siguen intactos, por eso se les conoce como cifradores de flujo síncronos.

El Cifrador Hélix al ser dependiente del contenido del texto a cifrar y su longitud, no sugiere un uso apropiado en canales físicos de transmisión o en ambientes susceptibles, luego, la pérdida de un dato compromete al resto. Se recomienda usar este cifrador en software, considerando este un ambiente estable, además, si se quiere usar el mensaje de autenticación, requería almacenamiento extra en el descifrador para validar los datos con los mensajes (el que recibe y genera).

La implementación de máquinas de estado para controlar cada proceso, permite ahorrar espacio y permite a la vez mayores velocidades al tener que usar todo bloque de construcción combinacional con un reloj común.

En comparación a los cifradores W7 y Hélix, A5/1 tiene un tamaño de clave pequeño, pues a pesar de inicializar el cifrador con 86 bits, solo 64 bits son la clave privada y 22 bits representa el número de trama, por lo que un ataque de fuerza bruta requiere 2^64 bytes, cantidad reducida comparando con las tecnologías computacionales disponible hoy en día.

El cifrador Hélix, tiene un bloque conformado de 20 rondas, que definiéndose de forma combinacional tiene un tiempo de respuesta mayor a los LFSR utilizados por A5/1 y W7.

El cifrador W7 requiere una máquina de estados con menos etapas que A5/1 y es 8 veces más rápido que este al hacer uso de celdas que operan paralelamente, donde cada una cifra un bit. Posee una restricción de utilización por clave de 2^46 bytes, reduciendo la frecuencia de cambio de claves, que en A5/1 se realiza cada 228 bits.

Los cifradores A5/1 y W7 son los candidatos más próximos a una implementación de hardware al hacer uso de operaciones XOR y desplazamiento de registros, que son ejercicios bastantes prácticos en hardware. Hélix en cambio añade suma aritmética, que en lógica booleana requiere un mayor uso de operaciones lógicas.

Una de las ventajas importantes de los sistemas de cifrado implementados, es la no exigencia de hardware distinto para el descifrado de datos, excepto a Hélix que implementa el mensaje de autenticación (MAC), lo cual requiere memoria en el descifrador para almacenar el MAC que recibe, y compararla con la que este genera con el fin de validar los datos.

El diseño de cifradores con los algoritmos propuestos garantiza privacidad de los datos y autenticidad para el caso del algoritmo Hélix. Sus respectivos diseños en VHDL permitieron una fácil implementación en la FPGA gracias al software Quartus II Web Edition, que compila los diseños, realiza análisis temporales y configura el dispositivo (FPGA) desde el programador.

VI. Referencias

[1] G. Gong, «Design of Stream Ciphers,» 2008. [En línea]. Available: http://comsec.uwaterloo.ca/~ece493t/A2.pdf..

[2] S. Thomas y D. Anthony, «The W7 Stream Cipher Algorithm,» IETF, Octubre 2002. [En línea]. Available: http://tools.ietf.org/html/draft-thomas-w7cipher-01

[3] N. Ferguson, D. Whiting, B. Schneier, J. Kelsey, S. Lucks y K. Tadayoshi, Helix Fast Encryption and Authentication in a Single Cryptographic Primitive, 2003.

[4] V. A. Pedroni, Circuit Design with VHDL, MIT Press, 2004.

[5] Altera, 2013. [En línea]. Available: www.altera.com.

[6] ECRYPT, «Notes on the ECRYPT Stream Cipher project (eSTREAM),» 2013. [En línea]. Available: http://cr.yp.to/streamciphers.html.

Deiby Y. Cañón M. Nació en Bogotá (Cundinamarca - Colombia) el 20 de Abril de 1990. Actualmente estudia en la Fundación Universitaria San Martín en el programa de Ingeniería en Telecomunicaciones.
Sus áreas de interés son los sistemas digitales y el procesamiento digital de señales.