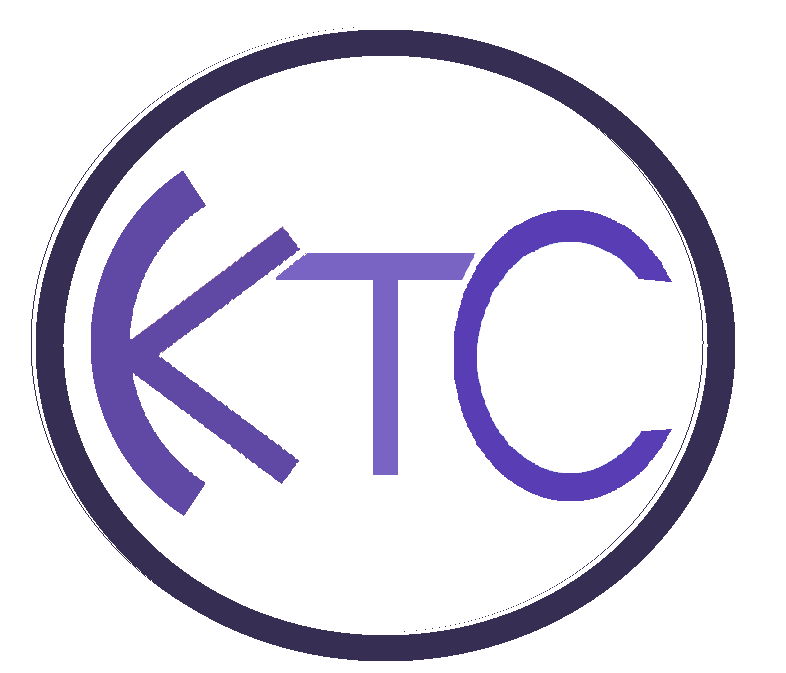
25 de mayo de 2018

Key Time Capsule

(Cápsula de clave temporal)



Miembros del equipo:

|  |  |
| --- | --- |
| Pardi, Leonardo Niels (JdP). | [ln.pardi@alumnos.upm.es](mailto:ln.pardi@alumnos.upm.es) |
| Asensio de León, Beatriz | beatriz.asensio.deleon[@alumnos.upm.es](mailto:ln.pardi@alumnos.upm.es) |
| Blanco López, Alejandro | [a.blancol@alumnos.upm.es](mailto:a.blancol@alumnos.upm.es) |
| Lozano Trujillo, Ismael | [ismael.lozano.trujillo@alumnos.upm.es](mailto:ismael.lozano.trujillo@alumnos.upm.es) |
| Méndez Fernández, Alejandro | [a.mendezf@alumnos.upm.es](mailto:a.mendezf@alumnos.upm.es) |
| Mesas Lafarga, Pablo. | [pablo.meses.lafarga@alumnos.upm.es](mailto:pablo.meses.lafarga@alumnos.upm.es) |

**Índice**

[Introducción 3](#_Toc515303659)

[Planteamiento del Proyecto 4](#_Toc515303660)

[Planteamientos propuestos 4](#_Toc515303661)

[Planteamiento elegido 6](#_Toc515303662)

[Fuentes 7](#_Toc515303663)

# Introducción

En la realización del PFA (Proyecto Final de Asignatura) se ha propuesto como idea para implementar en la FPGA un sistema de descifrado.

Este descifrador busca retener una clave durante un determinado periodo de tiempo a especificar por el usuario. Este sistema se basa en los estudios realizados por Ronald L. Rivest, Adi Shamir y David A. Wagner en el área de criptografía aplicada. La base para poder implementar este descifrador viene dada por la ecuación.

Siendo n un modulo formado por dos números primos muy grandes, t el parámetro donde se especifica el tiempo que en el que la clave quedará bloqueada, K es la clave para encontrar a partir de Ck que es la clave cifrada en origen. A, es un valor positivo aleatorio comprendido entre 2 y n.

Con esta premisa se ha desarrollado un pseudocódigo como paso previo al desarrollo del programa en VHDL que tendrá una aproximación hardware óptima compatible con la síntesis para FPGA, objetivo principal de dicho proyecto. A continuación, se muestra el código que se utilizará:

*= a*

*While t >0:*

*=*

*t = t -1*

*k = Ck –*

La motivación para desarrollar un proyecto de estas características viene dada por el cada vez mayor número de aplicaciones en las cuales se tiene en cuenta la seguridad. Como, por ejemplo, proteger información personal susceptible, restringir el acceso a determinados servicios (e.g control parental) o regular operaciones críticas en las que existe un intercambio de información confidencial.

# Planteamiento del Proyecto

En este apartado se explicará cómo se ha elegido la propuesta óptima para afrontar el desarrollo del proyecto. Tal y como se comentó en la sección anterior para la realización del PFA es necesario escribir un código previo en VHDL para su posterior implementación en la FPGA. Sin embargo, a la hora de seleccionar la mejor aproximación surgieron problemas de diversa índole que se detallarán a continuación.

## Planteamientos propuestos

En una primera fase, a modo de aproximación, se propusieron dos enfoques preliminares como posibles solución. Aunque fueron planteamientos teóricamente viables, en la práctica presentaron contratiempos considerables que llevo al equipo de desarrollo a desecharlos.

El primer planteamiento consistía en una transcripción literal del pseudocódigo, explicado en la introducción, a VHDL. Los dos problemas principales que tenía esta solución es: primero, que el programa que permite la configuración de la FPGA no permitía implementar el bucle puesto que el índice del *while* no era un valor fijo y, segundo, que el número de vueltas que da el bucle es superior a 2000.

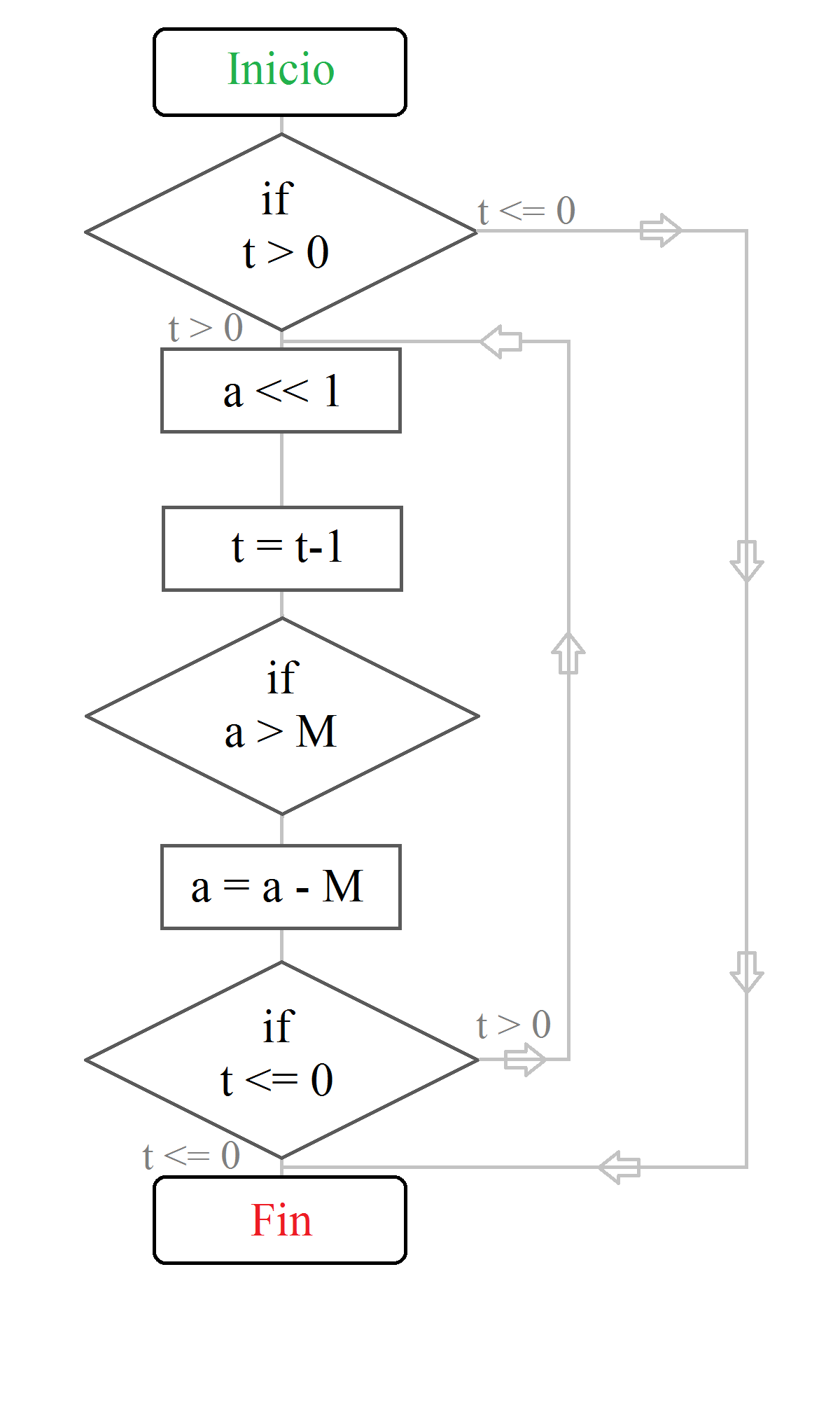


Figura 1. Algoritmo planteado

ALGORITMO DE ALEX

La segunda aproximación buscaba una orientación hardware. En concreto, se dividió el problema en diferentes circuitos lógicos, es decir, cada operación que debe realizar el sistema de descifrado equivalía a un determinado circuito lógico, véase multiplicadores, sumadores, etc. El principal problema que surgió es que el desarrollo adquirió una mayor dificultad. A continuación se muestra una figura explicativa del circuito inicial que se propuso como solución.

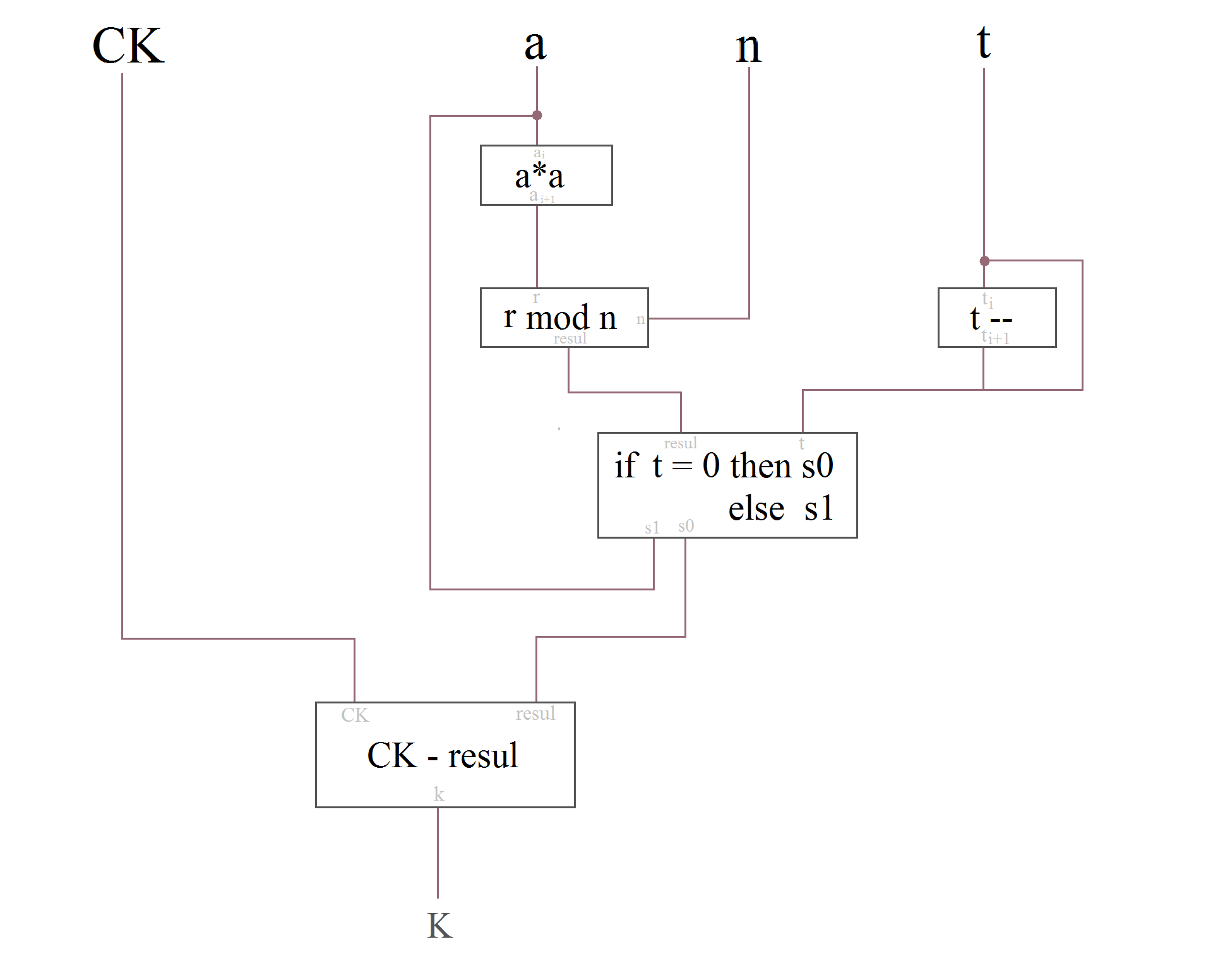


Figura 2. Circuito 2ª Aproximación 1

## Planteamiento elegido

Finalmente, tras varios intentos y propuestas fallidas se consiguió plantear una solución que permite afrontar la realización del sistema de descifrado.

La idea final que se ha considerado como válida para su posterior desarrollo consiste en una mejora del primer enfoque previamente explicado. Como el problema principal residía en el desarrollo del bucle, tanto por no tener un índice fijo como por dar más de 2000 vueltas en su ejecución, se ha optado por añadir un reloj externo.

Este reloj sustituirá al índice que se necesita para coordinar el desarrollo de un bucle de programación estándar. Esto significa que en cada flanco de subida de dicho reloj se van a procesar todas las operaciones que se realizaban en cada iteración del bucle original: se comprueba si el exponente es mayor que cero, en caso afirmativo se lleva a cabo la siguiente operación:

*=*

*t = t -1*

En caso contrario, es decir, que el exponente sea cero se sale del bucle y se continúa con la ejecución normal del programa.

# Fuentes

(2018). Key Time Capsule, PHR. Proyecto Final de Asignatura PHR. – Repositorios creado por nosotros.(Creado el 07 de 05 de 18) Obtenido de <https://github.com/PabloMesas/Key-Time-Capsule-PHR>

(2018) Programa de antivicio. Repositorio de Github. (13 de 05 de 18). Obtenido de <https://github.com/Leoniels/Programa-de-antivicio>

(2009).VHDL process. (10 de 05 de 18). Obtenido de <https://www.ics.uci.edu/~jmoorkan/vhdlref/process.html>

(2011). IEEE Standard VHDL Language Reference Manual - Redline. publisher not identified. Xilinx. (26 de 04 de 18). Obtenido de <http://www.xilinx.com>

(1996). Time-lo ck puzzles and timed-release Crypto - Ronald L. Rivest, Adi Shamir, and David A. Wagner. Berkeley EECS. (4 de 05 de 18). Obtenido de <http://people.eecs.berkeley.edu/~daw/papers/timelock.pd>

(2018).Introducción al Lenguaje de descripción Hardware VHDL. Obtenido de <http://www.iearobotics.com/tmp/seminarios-vhdl.pdf>

(2018)VHDL Guides. Obtenido de <https://www.ics.uci.edu/~jmoorkan/vhdlref/>

(2018) UART, Serial Port, RS-232 Interface. Obtenido de <https://www.nandland.com/vhdl/modules/module-uart-serial-port-rs232.html>

(2018) UART VHDL. Obtenido de <https://eewiki.net/pages/viewpage.action?pageId=59507062>