Universidad Politécnica de Madrid

Grado en Ingeniería Electrónica y Automática

Gestor de inicio de sesión mediante contraseña en VHDL

Grupo nº 12

Alberto Álvarez López (55117)
Sergio Berrendo Carretero (55147)
Luis Francisco Gallego Torres (55249)

ÍNDICE

INTRODUCCIÓN

Nuestro trabajo consiste en una variación del enunciado 2 propuesto en la oferta de trabajos VHDL. Este estriba en el diseño de un programa tipo “caja fuerte” mediante clave secreta que permite al usuario elegir su contraseña durante su primera vez en el programa. Posteriormente, el programa entraría en un estado de “Log in” donde debe insertarla de nuevo asegurándose de que es la misma que especificó al principio y, de ser así, podrá escoger entre cambiar la clave o ejecutar la funcionalidad del programa, que se muestra mediante el encendido de todos los LEDs de la placa. En caso de introducir una contraseña incorrecta, se le informa al usuario mediante un LED rojo y se vuelve a pedir la contraseña hasta que acierte.

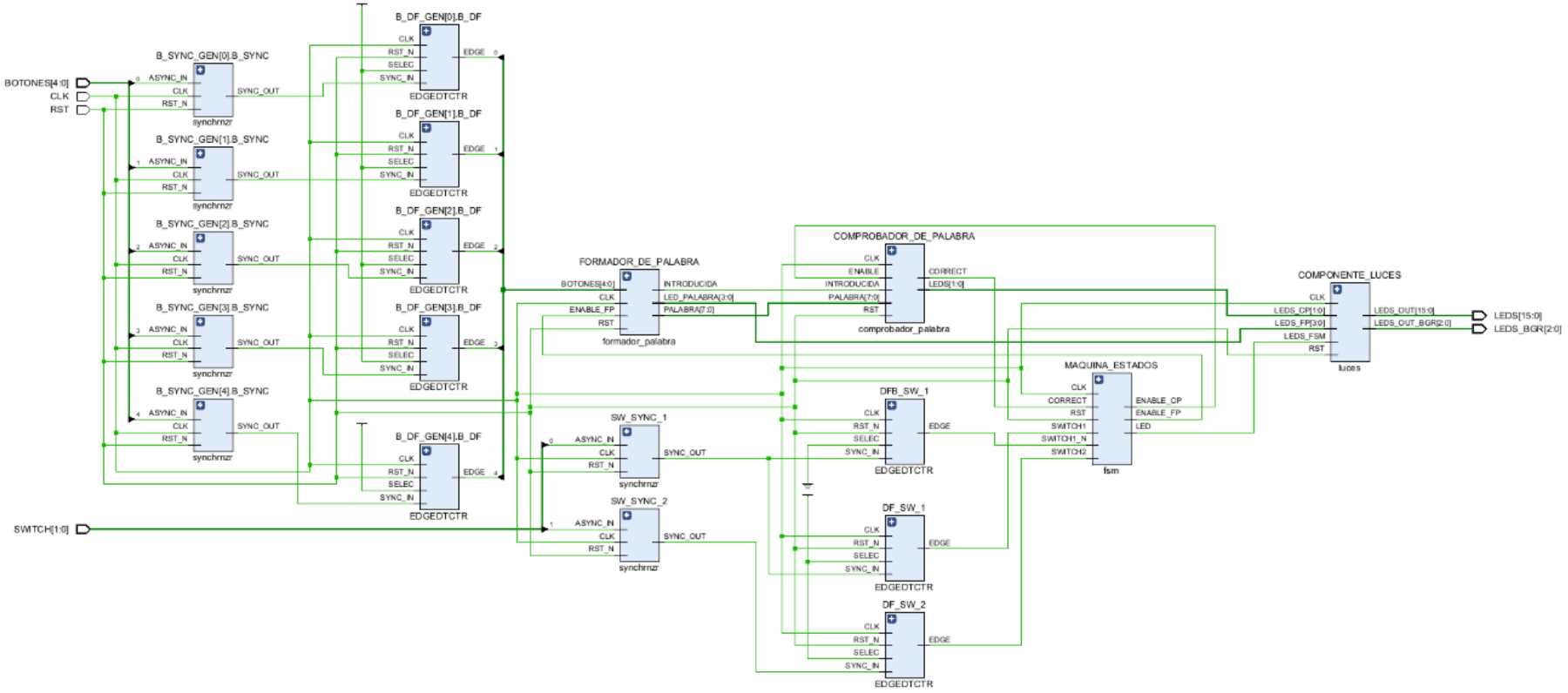
Estrategia y algoritmos desarrollados para la realización del trabajo

Desde un primer momento se optó por realizar el trabajo correspondiente a la clave secreta. Se plantearon diferentes diagramas secuenciales, que ilustraban las ideas que teníamos cada uno de los miembros, pero finalmente nos decantamos por llevar a cabo algo similar a un gestor de inicio de sesión actual, como el que se encontraría en cualquier página web, en el que se te pide inicialmente crear una contraseña y, a partir de ese momento, debes iniciar sesión con ella, pudiendo modificarla o realizar una funcionalidad del programa tras el inicio de sesión.

Una vez tuvimos el diagrama secuencial definitivo correspondiente a dicha idea, decidimos los componentes que serían necesarios para llevar a cabo el gestor y comenzamos a realizar su implementación. Como desde el primer momento planteamos adecuadamente cuál sería el funcionamiento del programa, no fue necesario realizar modificaciones significativas posteriormente, únicamente algún pequeño ajuste para cuadrar todo y que funcionase debidamente. Sin embargo, al principio tuvimos algún problema a la hora de implementar la idea, ya que no sabíamos cómo juzgar si era la primera vez que el usuario trataba de introducir una contraseña o intentaba cambiarla (función de registro), o si estaba intentando iniciar sesión y por tanto era preciso comprobar si la contraseña introducida correspondía con la correcta. A pesar de todo, fuimos perfeccionando nuestro algoritmo y acabamos logrando solventar dichos problemas mediante la adición de componentes encargados de realizar una función más abstracta, como pueden ser el formador o el comprobador de palabra, lo que ha supuesto que el programa termine cumpliendo nuestros objetivos de forma satisfactoria.

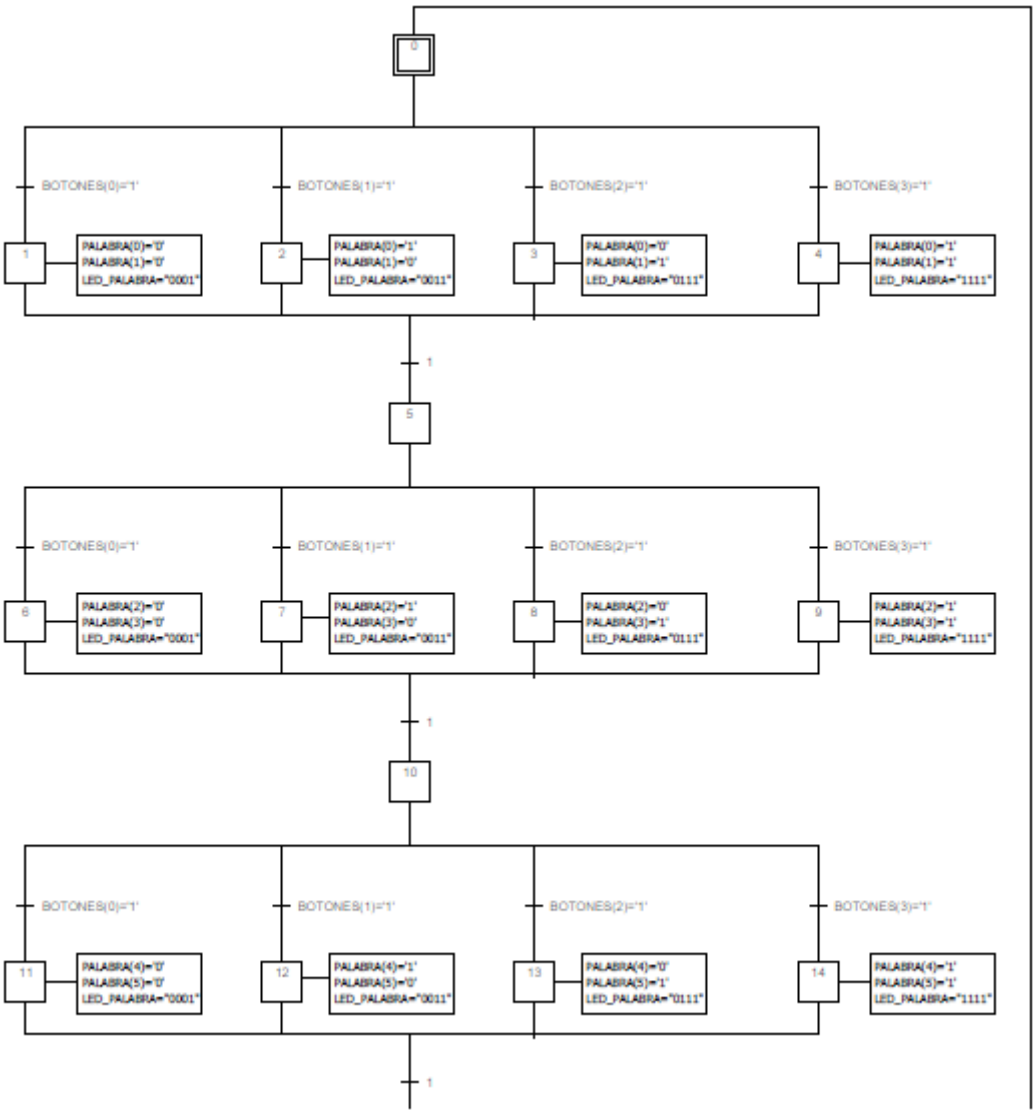
Funcionamiento de los bloques

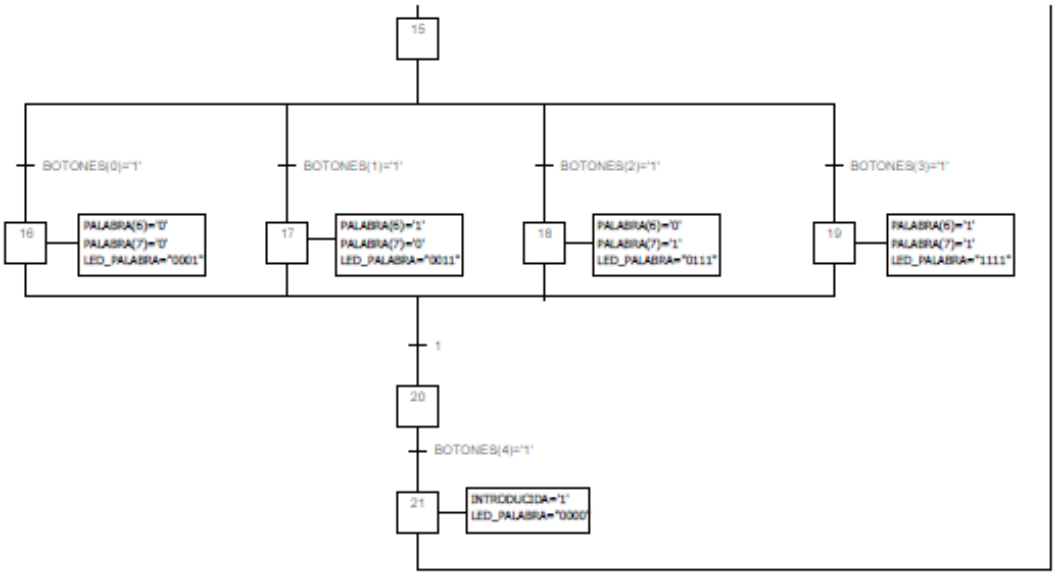
El diagrama de bloques de los componentes que forman nuestro proyecto es el siguiente:



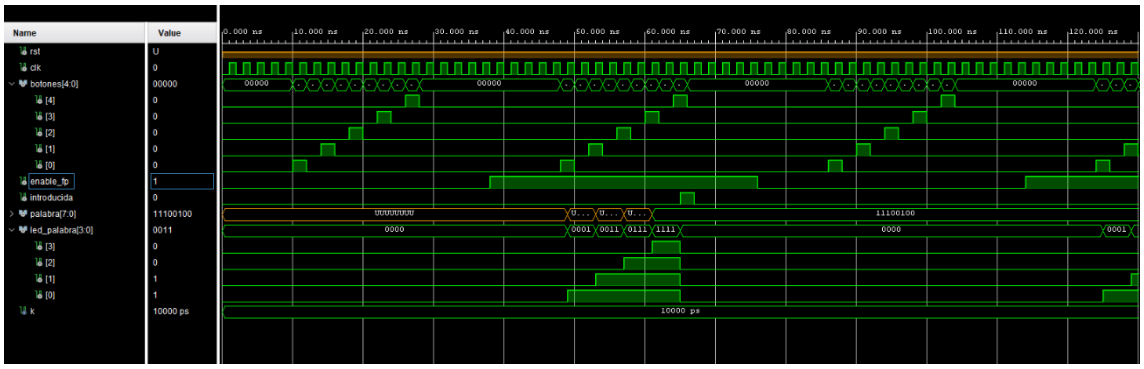
Funcionamiento de formador_palabra:

Su función es la de detectar los botones que presiona el usuario para formar una contraseña de cuatro dígitos. Inicialmente, el usuario selecciona un botón de entre los cuatro disponibles, se enciende el número de LEDs asignado a dicho botón para indicar que ha sido pulsado y dicho botón quedará registrado como primer dígito de la contraseña. Lo mismo sucede con los tres dígitos siguientes. Una vez registrada la clave en su totalidad, se debe presionar un pulsador de confirmación para avanzar en el programa, apagándose así los LEDs que indicaban el botón elegido.





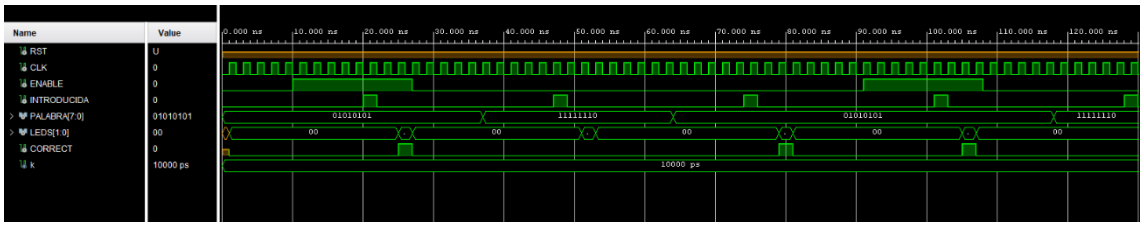
Se puede observar en el testbench inferior que, sin estar el enable activo, los botones no quedan registrados. Cuando se habilita la escritura, la clave se va actualizando conforme se pulsan los botones, y los LEDs se encienden con su botón correspondiente, y apagan al confirmar la contraseña.



Funcionamiento de comprobador_palabra:

Este componente cuenta con dos modos de trabajo: registro e inicio de sesión. Elije el modo de trabajo en función del valor de una señal llamada “enable_CP” que le envía la máquina de estados, en caso de valer ‘0’ (modo registro), la palabra que le llegue desde el formador de palabra se establece como la contraseña de acceso al programa para dicha persona y, además, se valida la contraseña enviando un pulso de la señal “correct”, necesario para avanzar de estado. Por otro lado, si “enable_CP” se encuentra desactivada, el componente procede a trabajar con el segundo modo de funcionamiento: modo “log in”. Este consiste en verificar que la clave o palabra introducida coincide con la establecida como contraseña en el programa. En caso afirmativo, se le informa al usuario a través de un LED de color verde y se le envía un pulso de la señal “correct” a la máquina de estados para que esta se encargue de avanzar a la siguiente etapa del programa. En caso contrario, se informa con el encendido del mismo LED mencionado anteriormente pero ahora de color rojo, además de no enviar el pulso necesario para cambiar de estado, forzando al usuario a tener que introducir una nueva palabra.

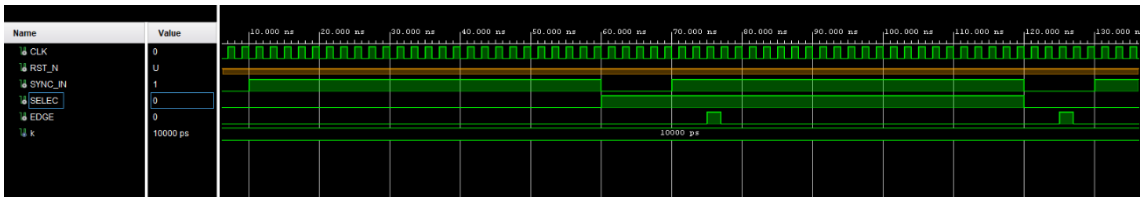
Como se puede comprobar en el testbench, el funcionamiento es el esperado:



Funcionamiento de edgedtctr:

El objetivo de este bloque es la detección de flancos que producen las señales para el correcto funcionamiento del programa. En concreto, su trabajo es generar un pulso limpio a raíz de una señal de entrada para asegurar que los demás componentes funcionan como se espera. El componente cuenta con un "requisito" para poder concluir que realmente ha habido un cambio intencionado a la entrada (se ha pulsado un botón o se ha cambiado el valor de un switch), si se hace suficientemente estricto este "requisito", el detector funcionará también como anti rebotes, en nuestro caso se valoró la posibilidad de incluir un componente extra con este fin pero se terminó por simplemente mejorar el detector de flancos. Además, se le ha añadido la posibilidad de trabajar como un detector de flancos de subida o de bajada en función del valor de la señal "select".

Se observa en el testbench que la detección de flancos de subida y bajada se realiza correctamente:

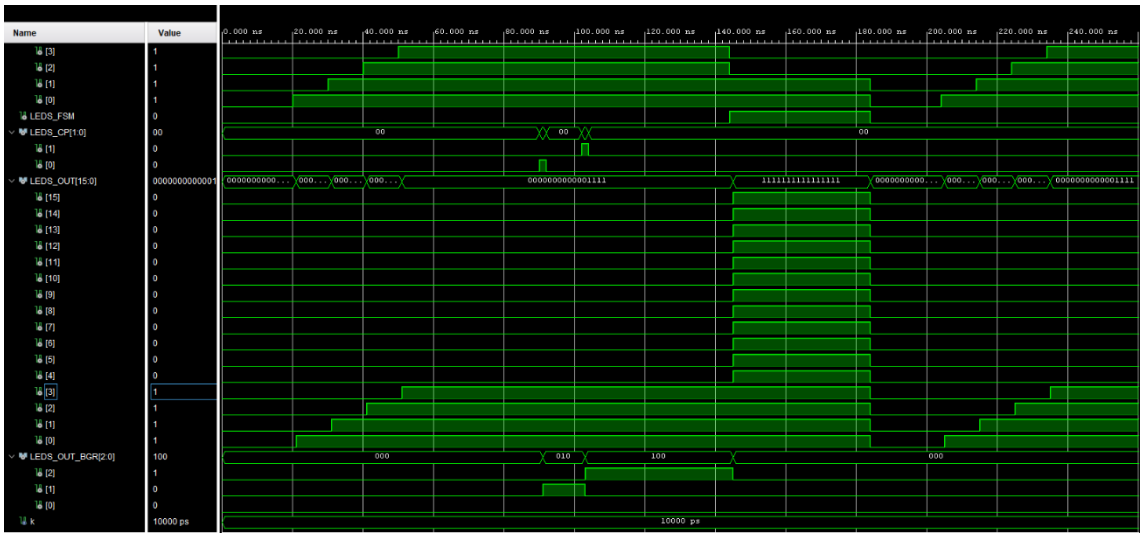


Funcionamiento de luces:

Es el encargado de gestionar todas las luces (LEDs de la placa) que se emplean en el proyecto en función de las señales que reciba de otros componentes. Específicamente, este componente se encarga de manejar las siguientes funciones:

- Enciende todos los LEDs durante la funcionalidad cuando la máquina de estados se lo indica.
- Enciende los LEDs que se usan como indicador del botón que ha sido presionado mientras se introduce la clave. El formador de palabra le informa directamente de qué botón ha sido pulsado.
- Gestiona el LED rgb incluido en la placa encendiéndolo con color verde o rojo para indicar si la contraseña introducida es correcta o no, según le informe el comprobador de palabra.

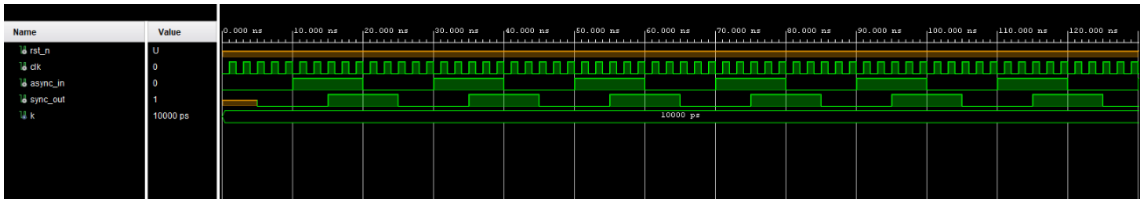
Se puede comprobar en el testbench que, en función de las señales que le lleguen, las luces se encienden y apagan como se espera:



Funcionamiento de synchrnzr:

Este componente, junto al detector de flancos, se encarga de asegurar que las señales que le llegan a los demás componentes son lo más adecuadas posible, para asegurar un funcionamiento más correcto y robusto del programa. El sincronizador se encarga en concreto de minimizar las posibilidades de que ocurra metaestabilidad. Esto ocurre cuando, por ejemplo, cambia la señal de entrada en alguno de los flancos de la señal de reloj sin respetar el tiempo de setup o hold.

Se realizó el siguiente testbench para comprobar su funcionamiento:



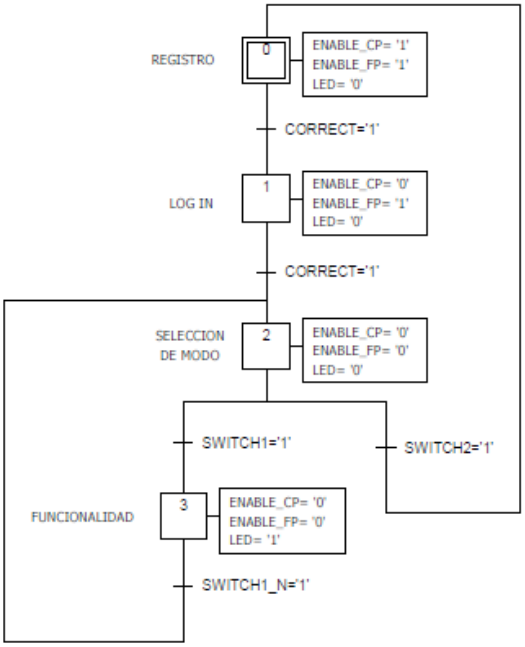
Funcionamiento de fsm:

Es el componente encargado de implementar la máquina de estados del programa. Cuenta con los siguientes estados:

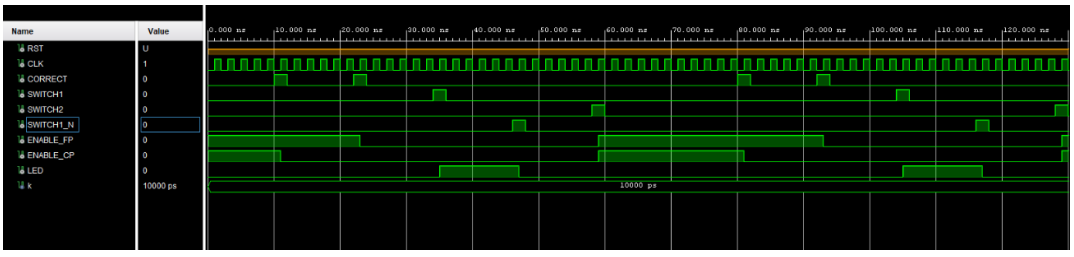
- Registro: durante este estado habilita al formador de palabra para recibir las pulsaciones de los botones (mediante “enable_fp”), le indica al comprobador de palabra que trabaje en modo registro y espera a que este componente le notifique de que ha sido introducida una contraseña para cambiar de estado.
- “Log in”: funciona de forma similar al estado anterior, con el único cambio de que ahora le dice al comprobador que trabaje en modo “log in”, por lo que no recibirá el pulso de “correct” para avanzar de estado hasta que la palabra introducida sea la correcta.

- Selección: en este estado se inhabilita al formador de palabra para que la pulsación de botones no interfiera en el programa y se espera al flanco positivo del primer o el segundo switch, sirviendo el primero para pasar al siguiente estado y el segundo para darle la posibilidad al usuario de cambiar la contraseña (devuelve el programa al estado de registro).
- Funcionalidad: se encarga de indicarle al componente luces que realice la funcionalidad, en nuestro caso hemos implementado una tan simple como el encendido de todos los leds, que sirve para mostrar de forma visual que hay acceso completo al programa, pero esta podría ser sustituida por cualquiera más compleja. Si durante estado se detecta un flanco negativo del mismo switch que provocó llegar a este estado, se vuelve al anterior, de tal forma que, una vez introducida la contraseña correcta de forma satisfactoria, se consigue libertad plena en el programa.

A continuación, se adjunta un modelo GRAFCET que representa el funcionamiento de la máquina de estados:



El testbench del componente muestra el comportamiento deseado:



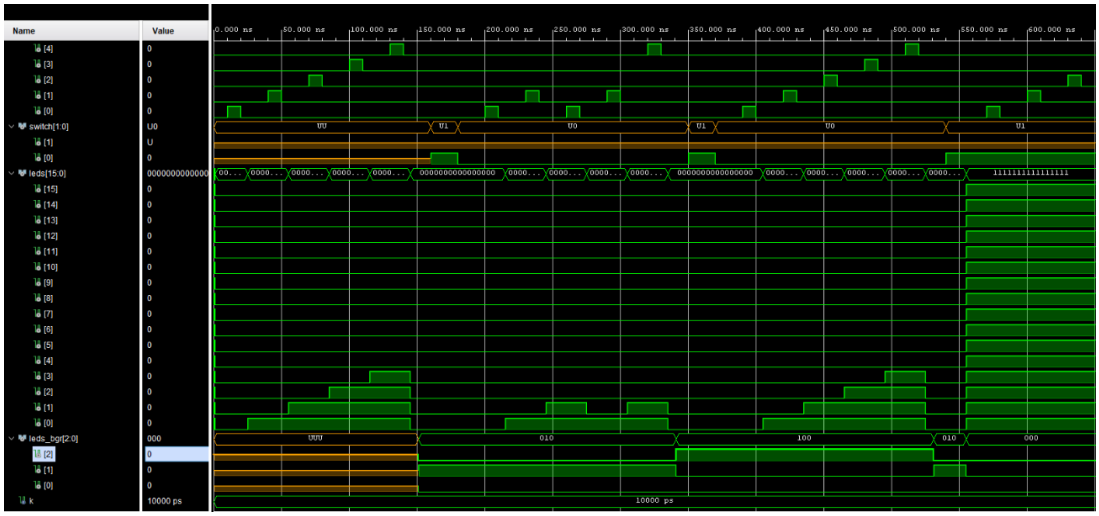
Funcionamiento del top:

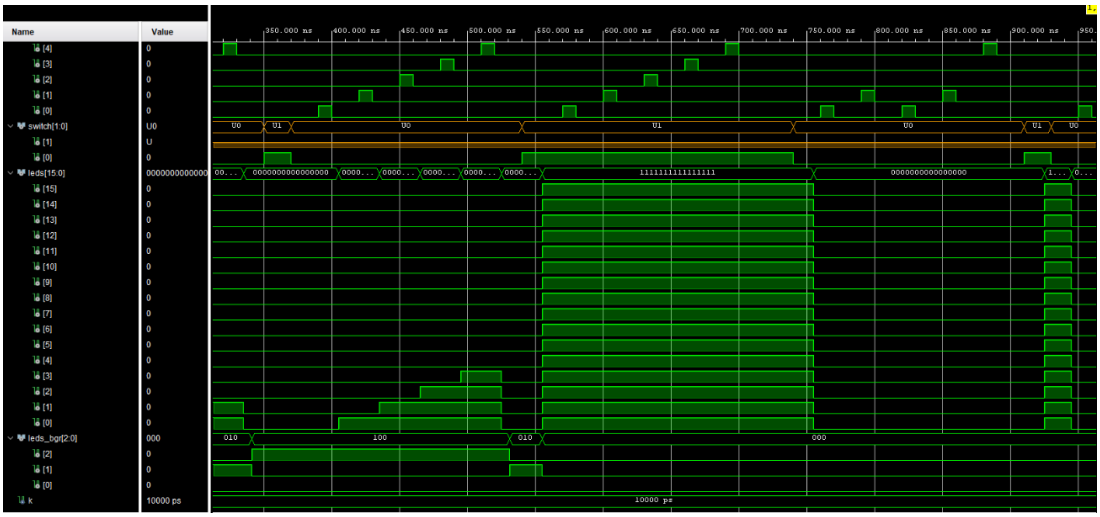
La finalidad de dicho elemento es la declaración de las entidades de todos los componentes que van a emplearse en la ejecución del programa, además de realizar todas las comunicaciones entre componentes a través de señales.

Para comprobar que todos los componentes interactúan correctamente, se ha realizado un testbench de la propia entidad top. A continuación, se muestra el efecto que tiene activar el reset global y el funcionamiento normal del programa:

- Funcionamiento global con Reset:

- Funcionamiento normal del programa:





CONCLUSIÓN

Tras la realización de este trabajo hemos adquirido fuertes conocimientos acerca de la escritura del lenguaje VHDL en FPGAs, como han sido el desenvolvernos con soltura a la hora de programar y de la creación de testbenches para comprobar que todo lo que hacíamos daba el resultado esperado. Además, pese a todos los errores y dificultades que hemos ido encontrando a lo largo del camino, hemos sabido dar buen uso de las herramientas disponibles y de nuestros conocimientos de la asignatura para poder solventarlas y sacar adelante un trabajo del que nos sentimos orgullosos. Por último pero no menos importante, cabe destacar el compañerismo y las ganas de trabajar de todos los miembros del equipo que hemos mostrado durante estos meses de trabajo.