SISTEMAS ELECTRÓNICOS DIGITALES

TRABAJO VHDL ASCENSOR DE 4 PISOS

GRUPO 1

Profesor: Pedro Luis Castedo Cepeda
Participantes:
Mario Sánchez García(55452)
Felipe de Gracia Ruiz (55199)
Adrian Juan Drag(55219)

ÍNDICE

Introducción

El primer trabajo de la asignatura consistirá en el diseño y elaboración de un ascensor de 4 plantas, implementado en la NEXYS 4 DDR y controlado por 5 botones, uno por cada piso y el último referido al RESET. Las salidas serán los LEDs y los displays, que se explicarán más adelante con mayor profundidad.
El funcionamiento del ascensor consistirá en la espera con las puertas abiertas hasta que el cliente pulse el botón al piso al que desea ir. Una vez pulsado, las puertas se cerrarán y el ascensor se pondrá en movimiento. Cuando esté en movimiento,debe hacer caso omiso a cualquier llamada a otro botón. Cuando llegue a su destino, este se parará y abrirá las puertas.

Desarrollo de la estructura del programa

Se tuvieron distintas ideas anteriores a la definitiva, teniendo en cuenta cómo encapsular los distintos bloques para su mejor funcionamiento. Finalmente se desarrolló la idea general de tener una entidad top con varios bloques más pequeños, entre ellos otras entidades top, siendo las más importantes la del control del ascensor por un lado y la cabina por otro, es decir, un bloque con toda la lógica basado en una máquina de estados que manejaría el funcionamiento del segundo bloque, que sería como nuestra maqueta de la cabina, con los motores del ascensor y el de la puerta.
Además, se crearían otros bloques como el antirrebotes, la salida de los LEDs y displays y el generador de pulsos.

Diagrama de bloques

La mejor manera de explicar el trabajo y la manera en la que está distribuido el diagrama de bloques es mediante un caso de uso.
Una persona entra a nuestro edificio y tiene que ir al 3er piso, por lo que prefiere coger el ascensor antes que subir por las escaleras. En ese momento no hay nadie utilizando el ascensor, y su posición actual es el piso 0, por tanto, nuestro cliente pulsará el botón 3.
En nuestra FPGA, se pulsaría el botón referido al 3er piso y en el diagrama de bloques, este botón es una entrada de la entidad top, que pasaría primero por el bloque antirrebote.

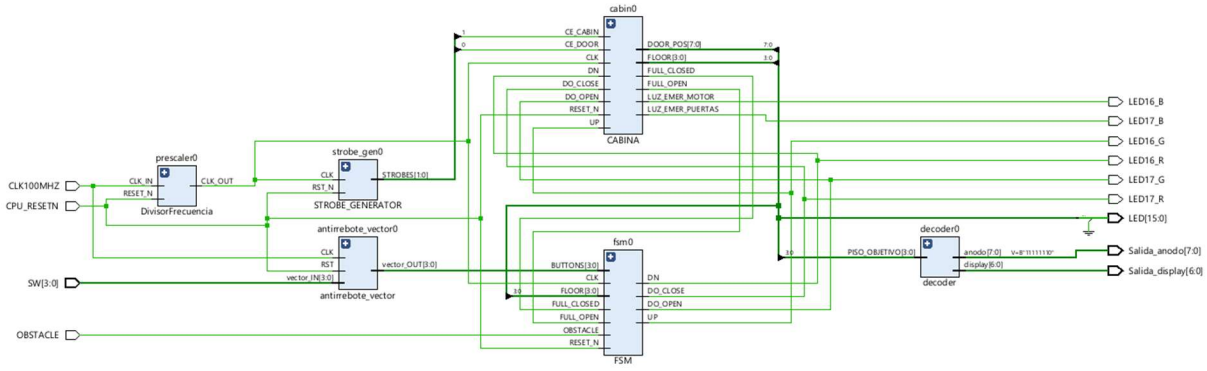

Ilustración 1: Entidad TOP

Bloque antirrebotes

Este bloque simplemente lo que hace es comprobar el cambio de estado del interruptor un determinado número de veces en un periodo de tiempo muy corto para asegurarse de que se ha presionado el botón y el cambio no ha sido debido al ruido.
Una vez se ha obtenido la señal de que un botón se ha pulsado, en este caso el 3, esta señal llega como entrada al bloque de control.

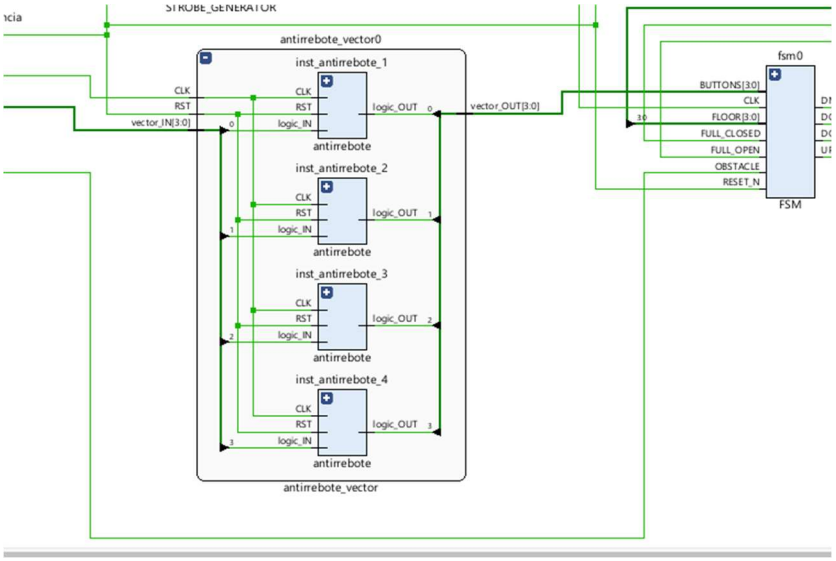

Ilustración 2: Entidad antirrebote_vector

Bloque de control

Este bloque de control se constituye de una máquina de estados, en concreto 5 (INICIAL, CERRANDO, SUBIENDO, BAJANDO y ABRIENDO). A este bloque le debe llegar la información de la cabina, es decir, los estados del motor del ascensor y el de la puerta y el piso actual en el que esté. Con esta información se pudo realizar la lógica responsable del cambio de estado mediante el uso de tres process. Si se pulsa el RESET se vuelve al estado INICIAL. Para pasar de este estado al estado de CERRANDO, se debe pulsar un botón distinto al piso en el que se encuentre la cabina. Una vez cerradas las puertas, se pasa a SUBIENDO o BAJANDO (comparando si el botón pulsado es mayor o menor al piso actual) hasta llegar al piso que se ha indicado, entonces se pasará al estado ABRIENDO. Una vez se hayan abierto las puertas, se pasará por último al estado de INICIAL, esperando a que se vuelva a pulsar algún botón. En cada estado están asignadas las salidas que debe tomar nuestra la cabina, es por ello que este bloque se llama de control, ya que se le indica a la cabina si sus accionadores deben estar encendidos o apagados.

Una vez explicado su funcionamiento, se sigue desarrollando el caso de uso. Como ha llegado la señal del 3er botón y la cabina le indica al control que el piso actual es el 0, se pasa al estado de CERRANDO y el accionador de la puerta se cierra y la máquina de estados tras realizar la comparación y que la puerta esté cerrada, indica que el siguiente estado va a ser el de SUBIENDO. Se enciende el accionador del motor del ascensor hasta llegar al piso indicado y se pasa al estado de ABRIENDO, y una vez abierta la puerta se pasa al estado de INICIAL con las puertas abiertas y el motor del ascensor parado.

Bloque de cabina

Dentro de este bloque se encuentran otros dos bloques a su vez, el **bloque del motor del ascensor** y el **bloque de las puertas**. A la cabina le llega la información del bloque de control que es quién actúa sobre los accionadores.
El bloque del motor de la cabina es el responsable de cambiar de piso por sí mismo y el bloque de las puertas es el responsable de abrir y cerrarlas, realizando un barrido de leds para indicar su estado.
Cuando se pulsa el botón para cambiar de piso, se puede observar cómo se abren y cierran las puertas mediante este barrido y se podrá ver en los RGB el estado del accionador del motor de la cabina.

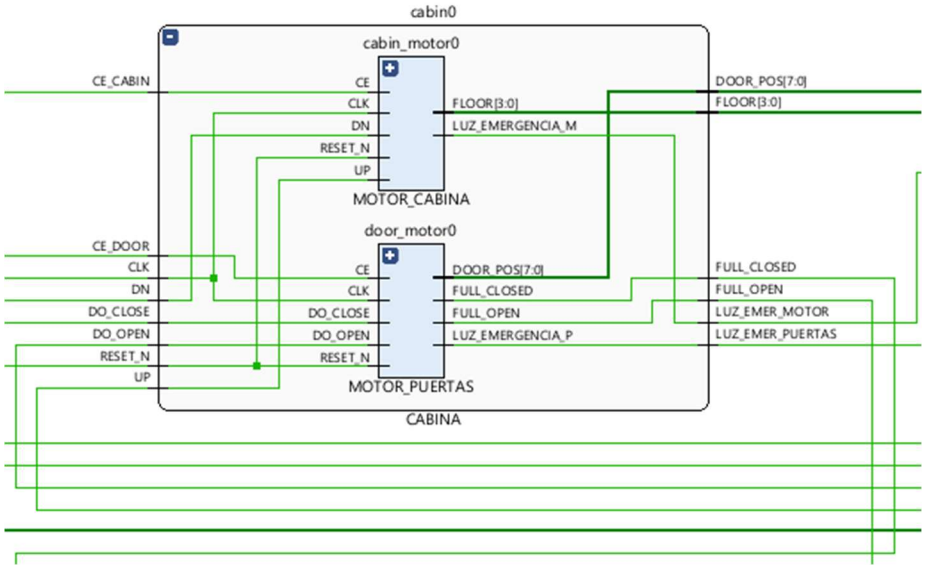

Ilustración 3: Entidad Cabina

Bloque decoder

Con la implementación de este bloque, se puede ver el piso en el que esté la cabina en ese momento en la salida de un display.

Bloque stober

El bloque strober es fundamental para el funcionamiento de nuestro programa ya que gracias a él la sincronización entre las entidades es fiable. Además, con esta entidad podemos visualizar lo que pasa en todo momento en nuestra placa porque se activaran o no las entradas ENABLE de cada entidad en función de cuando les toque.

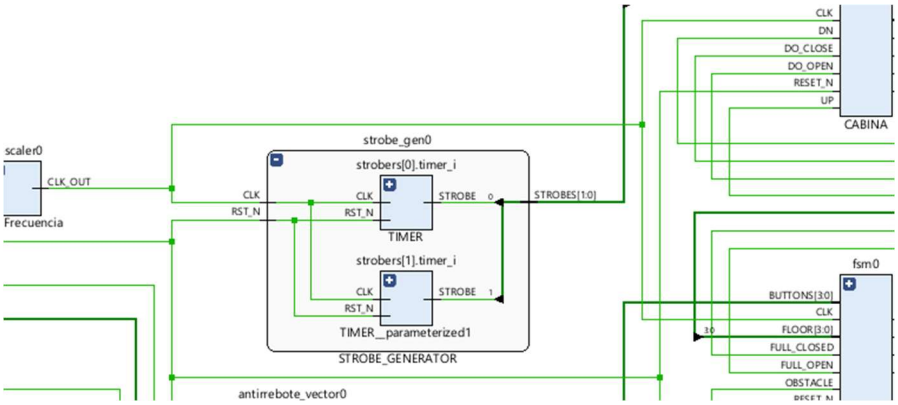

Ilustración 2:Entidad stober

Bloque divisor de frecuencia

Este bloque es el responsable de variar la frecuencia interna del reloj de la placa que en este caso es de 100MHz, cambiándola a una frecuencia menor que se decida para poder ver con detalle cómo el ascensor va pasando de un estado a otro, ya que si se utilizase la frecuencia interna no se vería el funcionamiento completo del proyecto, es decir, no se percibe cómo va cambiando de piso ni como se abren o se cierran las puertas, ya que los seres humanos no son capaces de percibir esos cambios a una frecuencia tan alta.

Problemas

Este proyecto no parecía difícil en un principio, ya que la lógica de programación básica de un ascensor no es tan complicada, pero desde un primer momento fueron surgiendo problemas. Lo primero que se hizo fue realizar el diseño de la estructura del código, cómo iban a estar conectados los distintos bloques, siendo una tarea más complicada de lo que parecía en un principio. Una vez tuviésemos nuestro diseño, (aunque luego se fue haciendo algún que otro cambio durante el desarrollo del trabajo) se codificaron los bloques y se realizaron las simulaciones tanto en Vivado como en EDA PLAYGROUND, comprobando cada bloque por separado para ver que todo estuviese bien. Una vez realizado esto, se empezó a probar en la placa y no funcionó como lo esperado, por lo que se tuvo que depurar el código, y solucionar los problemas que nos aparecían. El problema principal que tuvimos fue que todas las entidades si necesitan el reloj en su propio programa tienen que trabajar con el CLK, al suceder esto, nuestro programa funcionaba, pero iba tan rápido que no pudiamos apreciar por los pisos que pasaba. Finalmente, con la entidad strober pudimos solucionar el problema y hacer que cada entidad que quisiéramos funcionase con tiempos distintos y finalmente completar el trabajo.

Testbenchs

Los testbenchs nos han resultado muy útiles ya que cada vez que terminábamos una entidad podíamos comprobar mediante la simulación de EDA PLAYGROUND y VIVADO si el resultado era optimo.

Motor puertas

El testbench realizado de motor puertas nos puso varios problemas ya que tuvimos que juntar el barrido de leds dentro de esta entidad y coordinar los tiempos no fue una tarea fácil. Finalmente se puede comprobar que cuando se cierra una puerta ocurre dicho barrido.

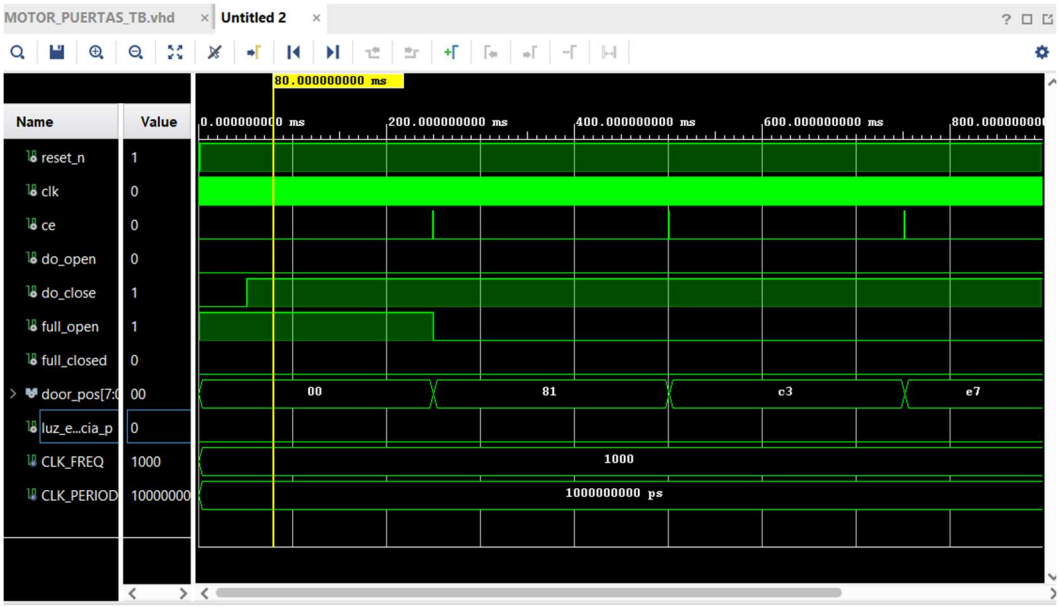

Ilustración 5: Testbench motor puertas

Comparador

La entidad comparador se encuentra dentro de entidad de motor_ascensor y es la que se encarga de saber en que posición se encuentra dependiendo de los avisos que le llegan de nuestro control. Podemos comprobar que cuando la señal de Aviso_ascensor es 2 (subir), el piso actual va cambiando su valor.

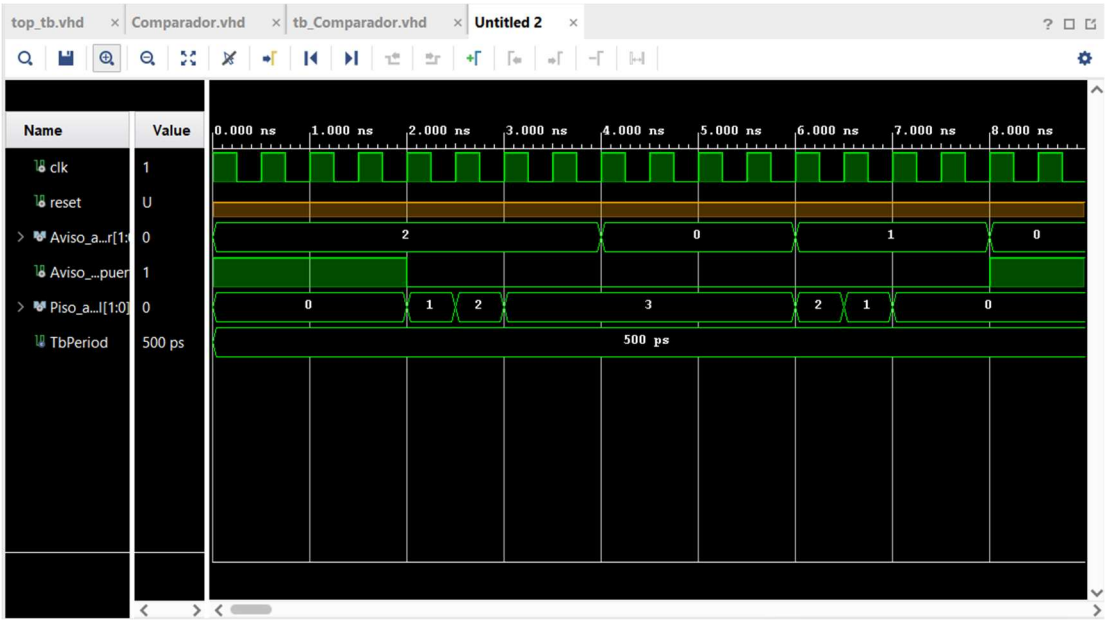

Ilustración 6: Testbench comparador

Barrido de led

Tanto el comparador como el barrido de Led es de las partes más importantes del trabajo porque gracias a ellos conseguimos que nuestro trabajado este totalmente automatizado, por ello tuvimos que comprobar el barrido de Led fuera de la entidad Motor_puertas. Finalmente se puede comprobar como cuando el Aviso_Puertas es 0, se van concatenando 1bit en cada flanco de reloj.

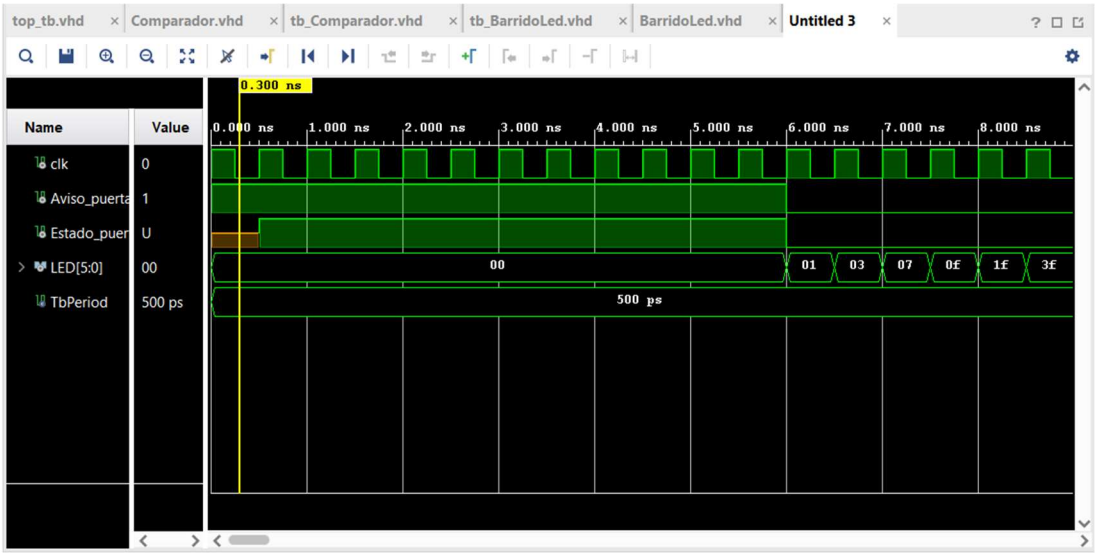

Ilustración 7: Testbench Barrido de Led

Motor movimiento

Esta es la salida de nuestros actuadores o en nuestros casos de los leds que se encienden en nuestra FPGA. Si el Aviso_ascensor es 1(bajar) nuestro Movimiento es bajar.

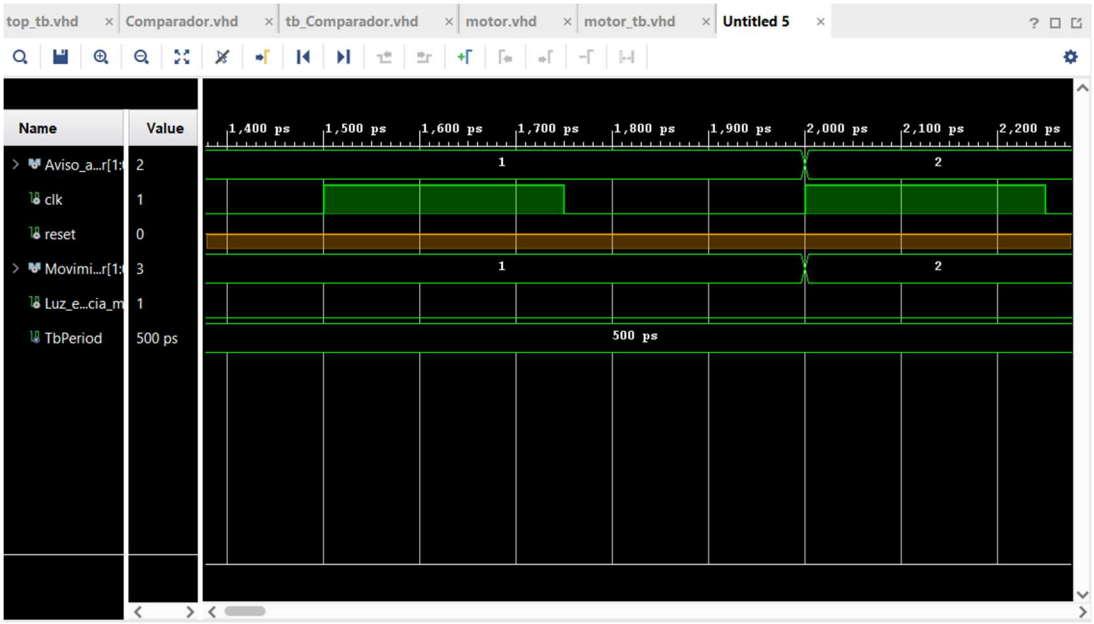

Ilustración 8: Testbench motor movimiento

Decoder

Nuestro decoder como se ha explicado anteriormente tiene que convertir a los valores al display. Se puede comprobar que cuando el piso Actual es el 0, nuestro decoder imprime el valor 7e (111 1110) se encienden todos los displays menos el del centro.

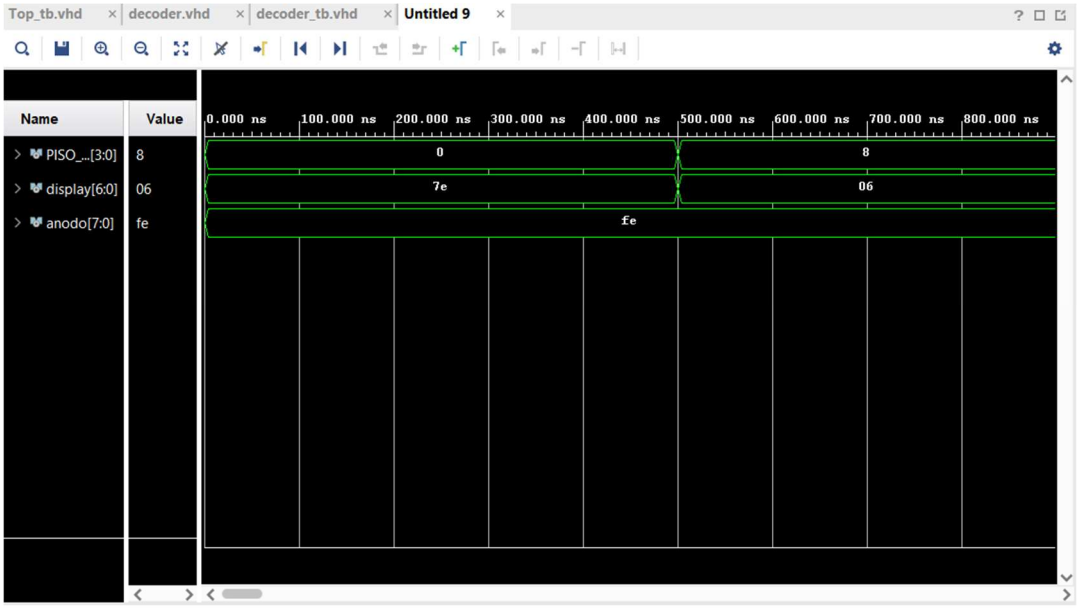

Ilustración 9: Testbench Decoder

Top

Por último, la entidad mas importante de todas, ya que gracias a ella conseguimos que nuestro ascensor este totalmente encapsulado. Para comprobar este testbench nos fijamos principalmente en las salidas de los displays y los motores. Podemos comprobar que cuando se introduce un piso, el barrido de leds se ejecuta y nuestros leds (RGB) se encienden.

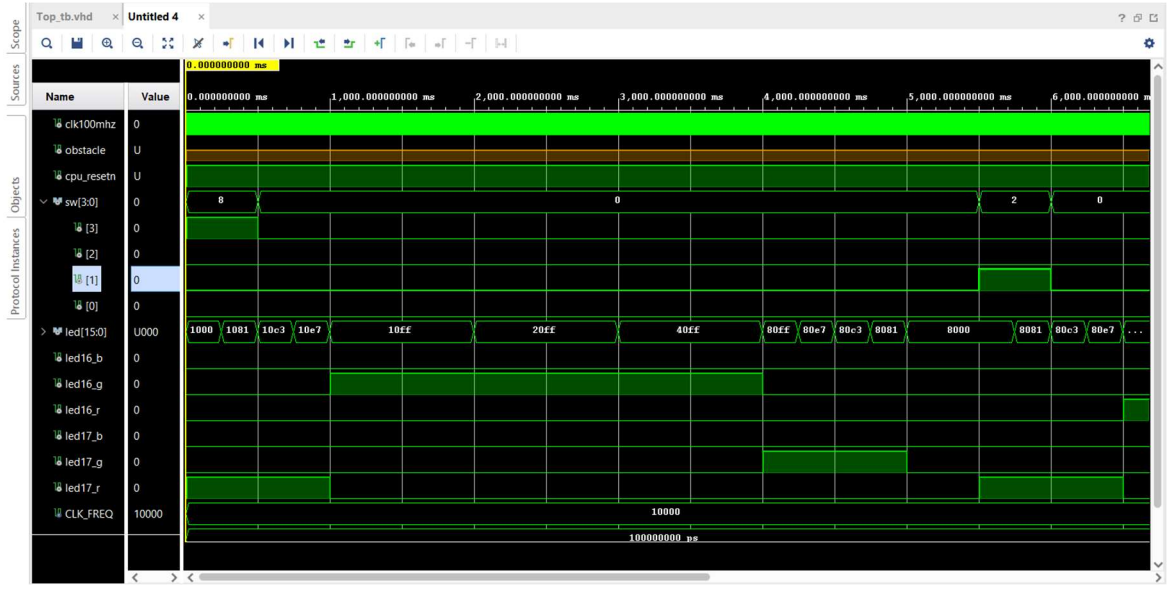

Ilustración 10: Testbench TOP

Repositorio de github

https://github.com/mariooot13/Trabajo-VHDL-SED-G1

Conclusión

El proyecto realizado ha resultado ser un gran aprendizaje en todas las fases de este. Se comenzó con la idea de tener un diagrama de bloques correcto de forma que la programación e integración de estos bloques fuese óptima.
Después de varias iteraciones y todo el trabajo empeñado se ha conseguido el objetivo inicial del proyecto que consistía en simular el funcionamiento de un ascensor de cuatro pisos con la interacción continua de los botones, switches, LEDS y display.
Se ha logrado una gran familiarización tanto con el entorno de trabajo Vivado como con la FPGA y el lenguaje VHDL, por lo que a medida que avanzó el proyecto se fueron intentando mayores complejidades para su adecuado funcionamiento.
Además, se han realizado pruebas tanto en la propia FPGA como con diversos testbench viendo el comportamiento del ascensor en cada etapa nueva ya que se ha ido de un menor nivel de abstracción a uno mayor finalmente.
En conclusión, el ascensor de cuatro plantas creado simula un ascensor real y se ha podido observar que la actividad de este es la adecuada.