UNIVERSIDAD POLITÉCNICA DE MADRID
ESCUELA TÉCNICA SUPERIOR DE
INGENIERÍA Y DISEÑO INDUSTRIAL
Ronda de Valencia, 3 - 28012 Madrid

TRABAJO VHDL
ASCENSOR 4 NIVELES

GRUPO 10

Alejandro Mayor Escalada – 54082
Lucía Pardo Hermosa – 55396
Roberto Vázquez Magdaleno – 53444

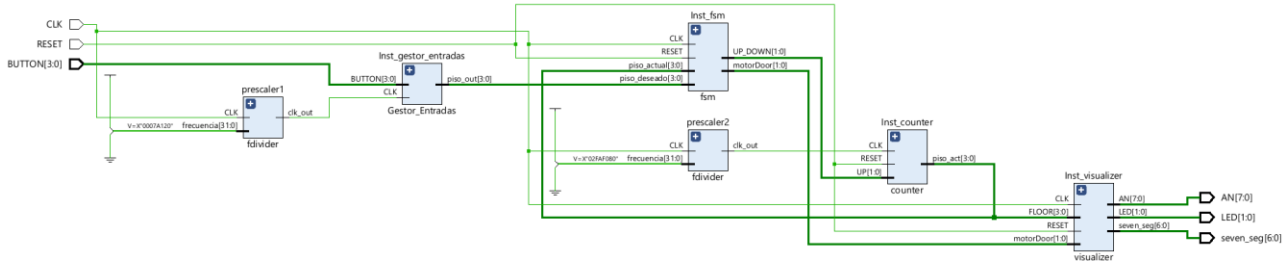
1| DESCRIPCIÓN DEL PROYECTO

Para el trabajo de VHDL, se ha realizado el control de un ascensor para una vivienda de 4 niveles. Emplearemos 5 botones como entradas, uno por cada piso y el botón de RESET. Para las salidas, utilizaremos unos LEDs, para indicar que las puertas se encuentran completamente abiertas o cerradas, y Displays para indicar que las puertas se están abriendo o cerrando, así como la planta en la que nos encontramos.

El funcionamiento consiste en pulsar el botón de la planta a la que se desea ir, en ese momento se cerrarán las puertas del ascensor (inicialmente las puertas se encontrarán abiertas), una vez llegado a la planta marcada, se abrirán las puertas y se mantendrán así hasta que se reciba otra llamada. Si se pulsa algún botón mientras se esté realizando la acción de subir o bajar, se ignorará esta llamada.

2| ESTRUCTURA DEL PROYECTO

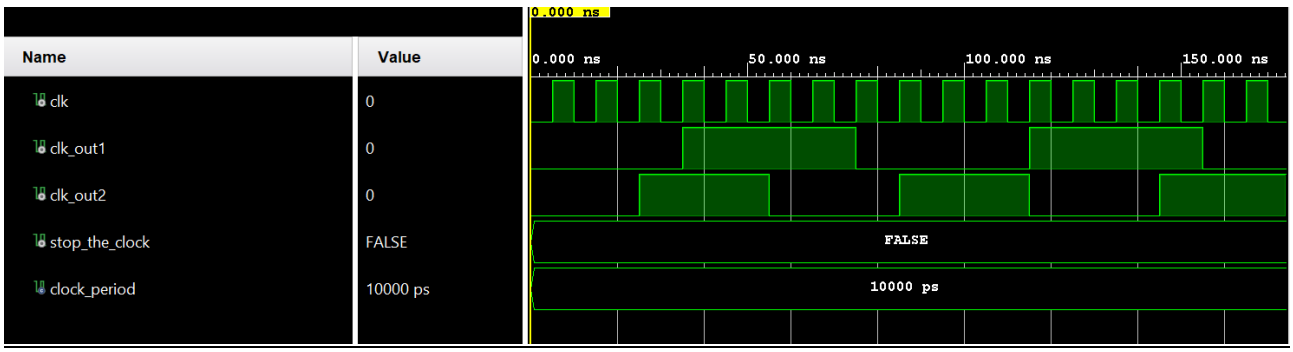
Se ha organizado el proyecto en una entidad *top* que engloba a otras entidades, cada una de ellas enfocadas en realizar acciones concretas.



2.1 DIVISOR DE FRECUENCIA

Puesto que la frecuencia de la FPGA resulta demasiado elevada para lo que requieren algunos componentes, se ha creado un divisor de frecuencia con el fin de obtener un tren de pulsos adecuado en función de la frecuencia que requiere cada componente para funcionar correctamente. Para ello, se ha creado una variable inicializada a cero que se incremente en una unidad por cada flanco positivo del reloj. De esta forma, mantiene la señal auxiliar del reloj a nivel alto hasta que el contador alcance el valor de la frecuencia deseada, generando un nuevo tren de pulsos.

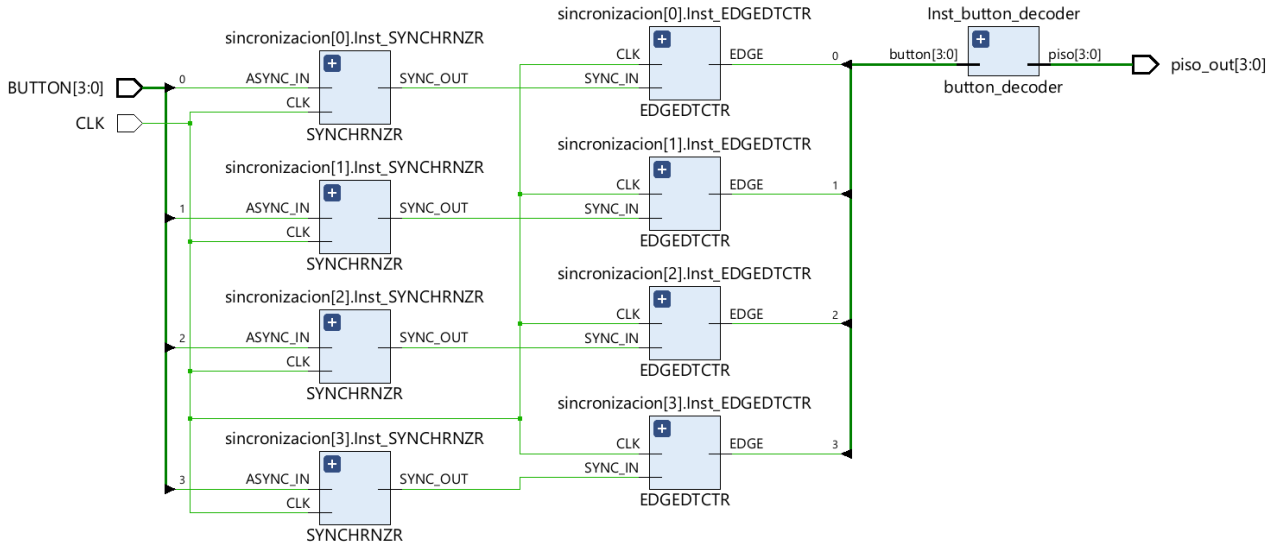
TESTBENCH



Se observa como para distintas frecuencias, se generan diferentes formas de onda del reloj.

2.2 GESTOR DE ENTRADAS

Se ha decidido incorporar una entidad encargada de gestionar las entradas, en este caso botones. Debido a problemas surgidos durante la implementación en la FPGA, finalmente se decidió incluir una frecuencia dividida del pulso de reloj para evitar pérdidas de información y detectar correctamente la pulsación de los botones.



2.2.1 SINCRONIZADOR

El principal problema que acarrea el uso de pulsadores son los rebotes o *bouncing*. Al ser accionados, los pulsadores tardan cierto tiempo en alcanzar un estado estable. Además, las entradas externas introducidas al sistema son una fuente de inestabilidades (metaestabilidades). Por lo tanto, para solucionar dichos problemas se utilizará un sincronizador.

Para evitar la generación de rebotes, suele ser necesario añadir un módulo antirrebotes, sin embargo, con la placa utilizada, el sincronizador es suficiente.

2.2.2 DETECTOR DE FLANCOS

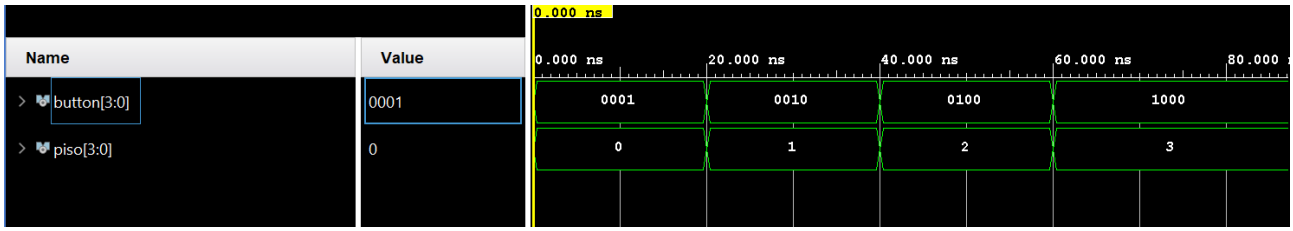
Una vez estabilizada la señal de entrada, el detector de flancos cambia a nivel alto si se ha pulsado un botón.

2.2.3 DECODIFICADOR DE BOTONES

Para trabajar directamente con los números de los pisos, se ha asignado a cada botón un piso.

BOTÓN	PISO
“0001”	“0000” (0)
“0010”	“0001” (1)
“0100”	“0010” (2)
“1000”	“0011” (3)

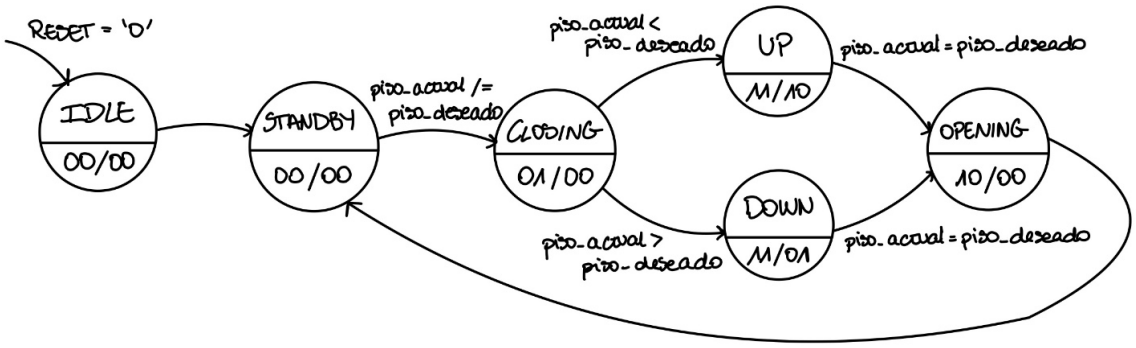
TESTBENCH



2.3 FSM

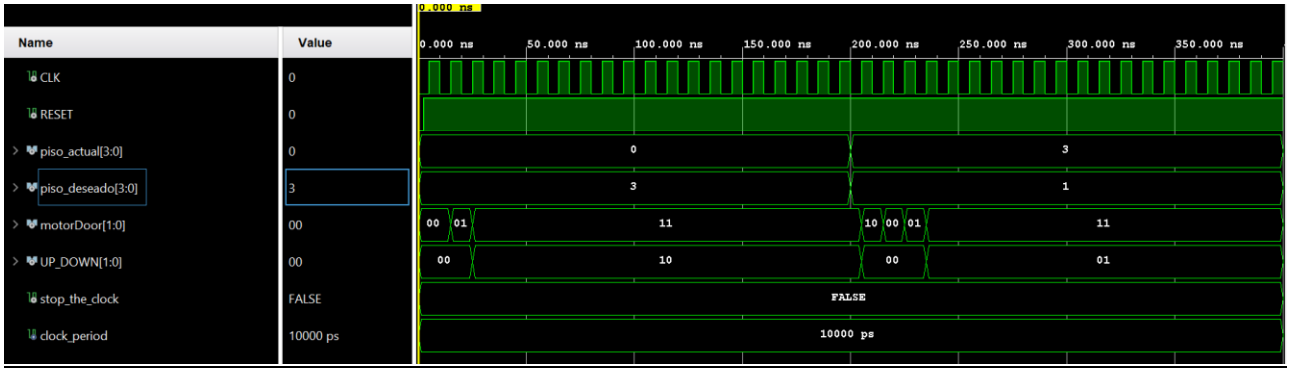
La máquina de estados está formada por 6 estados: IDLE, STANDBY, CLOSING, UP, DOWN y OPENING. Recibe como entradas la señal de reloj, el estado del botón RESET, el piso deseado y el piso actual. Las salidas son el estado del motor de las puertas y del motor del ascensor.

El estado inicial y al que se acude en caso de que se pulse RESET es IDLE, donde se activan los valores por defecto de las salidas (puertas abiertas y planta baja). Inmediatamente después, se salta al estado STANDBY, a la espera de la llegada de una entrada. Si recibe una llamada, donde el piso deseado es distinto al piso donde nos encontramos, cambiamos al estado CLOSING y se almacena el valor del piso al que se desea ir. En el estado CLOSING, se activa el motor de las puertas y se comprueba el piso deseado. Una vez detenido el motor de las puertas, si el piso deseado es mayor que el piso actual, se cambia al estado UP, en caso contrario, se cambia a DOWN. Los estados UP y DOWN se pueden analizar conjuntamente. Se activa el motor del ascensor y, una vez llegado al piso deseado, se cambia al estado OPENING donde se detiene el motor del ascensor y se activa el motor de las puertas. Finalizada esta secuencia, se regresa al estado de STANDBY a la espera de una nueva llamada.



motorDoor	UP_DOWN
00 – puertas abiertas	00 – standby
01 – puertas cerrándose	01 – bajar
10 – puertas abriéndose	10 – subir
11 – puertas cerradas	11 – error

TESTBENCH

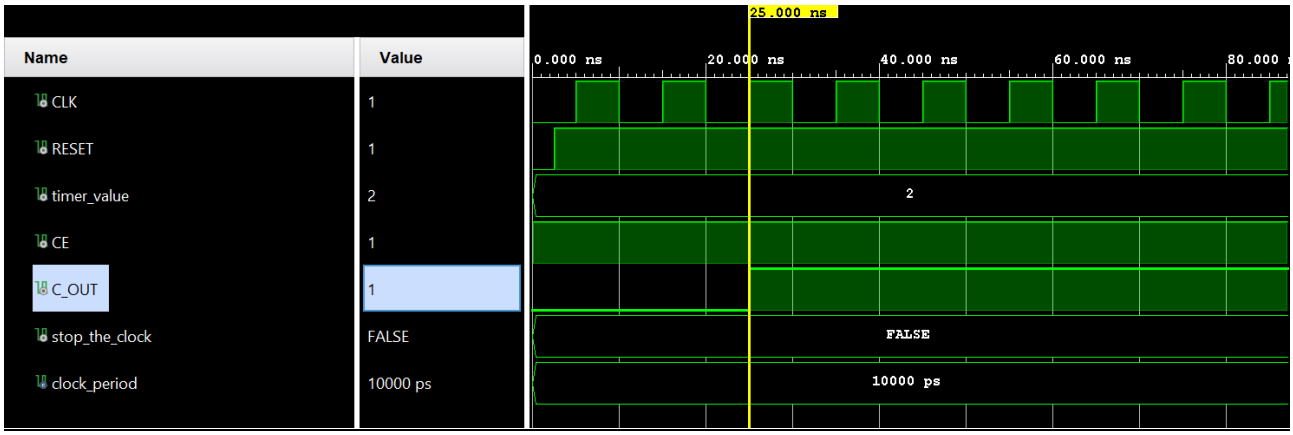


Observamos como la secuencia de estados de la fsm inicia con las puertas abiertas (“00”) y el motor del ascensor parado (“00”). Si nos encontramos inicialmente en el piso 0 y se pulsa el piso 3, el estado de las puertas cambia a cerrándose (“01”), mientras que el motor del ascensor se mantiene parado. Una vez cerradas las puertas, se activa el motor, en este caso para ejecutar la acción de subir (“10”). Una vez llegado al piso deseado, se detiene el motor y comienzan a abrirse las puertas (“10”). No se aceptará una nueva llamada hasta que las puertas se encuentren completamente abiertas. Lo mismo ocurre para la acción de bajar.

2.3.1 TIMER

Debido a que el cambio de estado en la fsm se produce a gran velocidad, se ha incluido una espera mediante un temporizador. La lógica del temporizador es similar a la del divisor de frecuencia, sin embargo, en este caso el temporizador se activa mediante un *‘chip enable’*, que inicia una cuenta hasta llegar al valor deseado de espera, finalizada esa espera se activa una señal de salida que da paso al siguiente estado de la fsm.

TESTBENCH



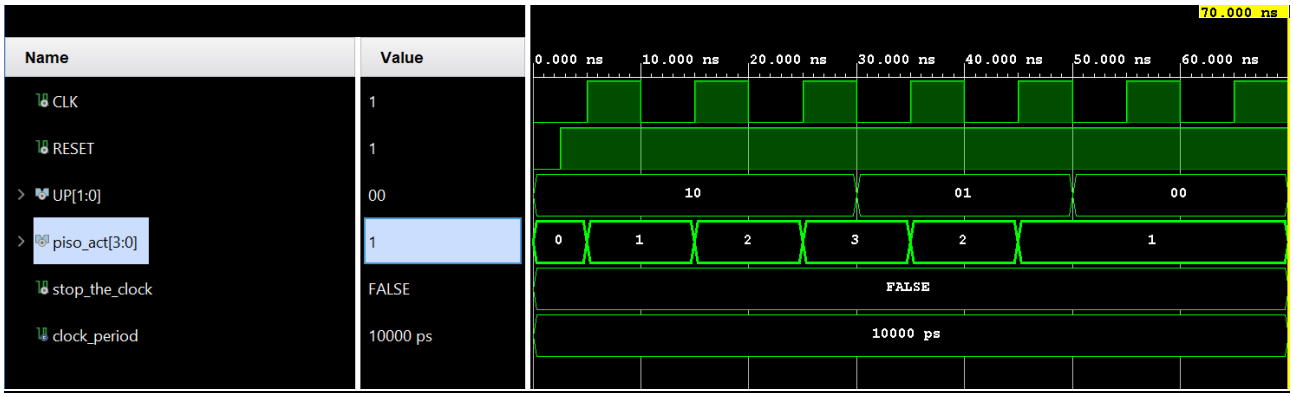
Al incrementar en una unidad el valor del contador, observamos que, una vez llegado al valor de espera del temporizador, en este caso dos, el valor de la señal de salida se activa.

2.4 CONTADOR

Para realizar el cambio de piso, se ha utilizado un contador. El estado del motor del ascensor que sale de la máquina de estados se emplea como entrada en el contador. De esta forma, si el ascensor sube, el contador se incrementará hasta alcanzar el piso deseado. Por otro lado, si el ascensor está bajando, el contador decrementará el valor de los pisos. La salida de este bloque se corresponde con el piso en el que nos encontramos, que se realimenta con el componente de la fsm.

En este bloque se ha incluido un divisor de frecuencia para que resulte visible el cambio de piso.

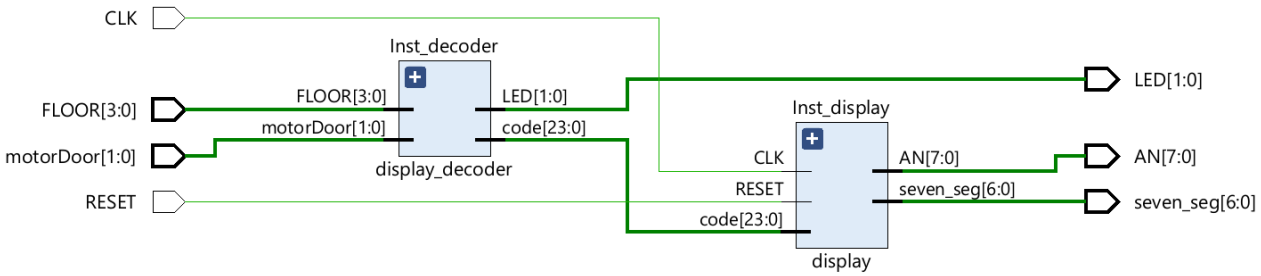
TESTBENCH



Observamos que cuando la señal del motor del ascensor está en “10” (UP), se incrementa el contador, mientras que si está en “01” (DOWN), se decrementa. Por otro lado, si el valor es “00” (STANDBY), se mantiene el valor del último piso en el que se encontraba.

2.5 VISUALIZER

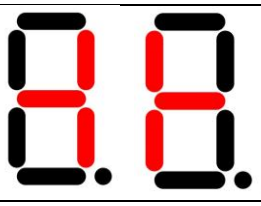
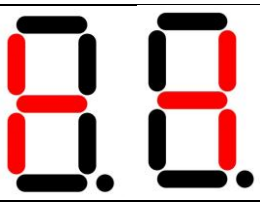
Para gestionar las salidas de los displays y los LEDs, se creará un bloque para dicha gestión.



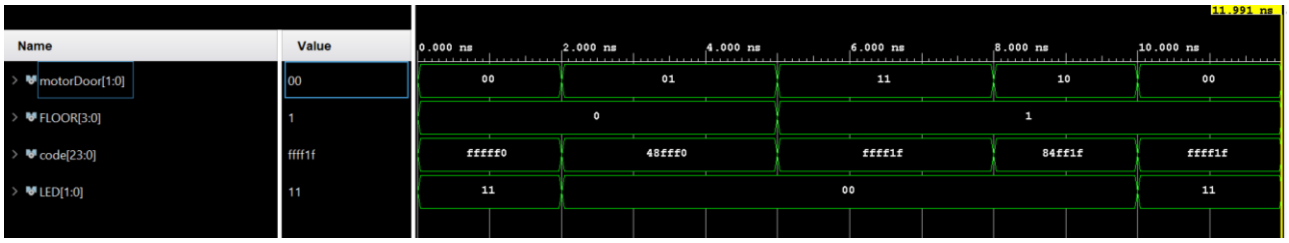
2.5.1 DECODIFICADOR DE DISPLAYS

Puesto que se quieren mostrar dos datos distintos en los displays, se ha incluido una señal intermedia que almacene los datos que se quieren mostrar, en función del display al que le corresponda esa información.

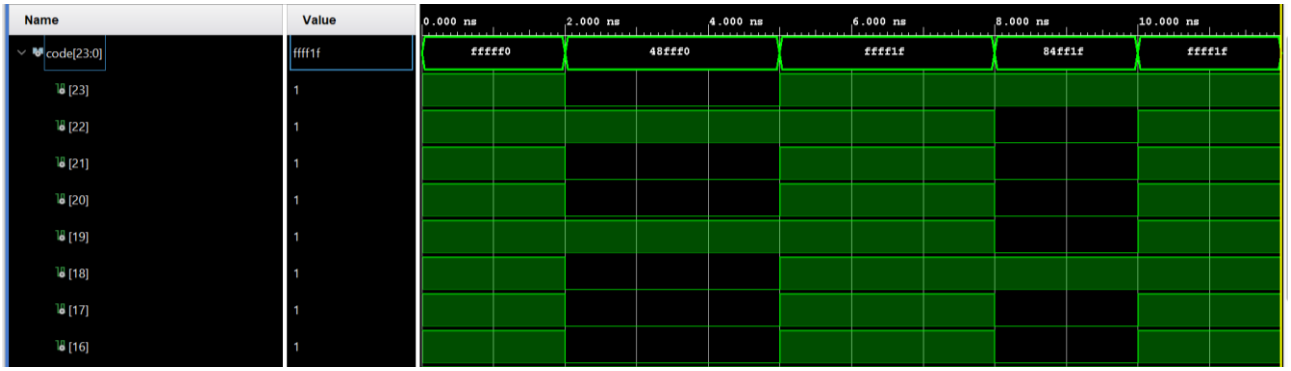
Se emplearán cuatro displays para mostrar los pisos, uno por cada piso, y dos displays para mostrar el estado en el que se encuentran las puertas (abriéndose o cerrándose). Por lo tanto, se necesitan 4 bits por cada display utilizado, es decir, 24 bits. Se explicará más detalladamente con ayuda del testbench.

PUERTAS CERRÁNDOSE	PUERTAS ABRIÉNDOSE
“01001000”	“10000100”
	

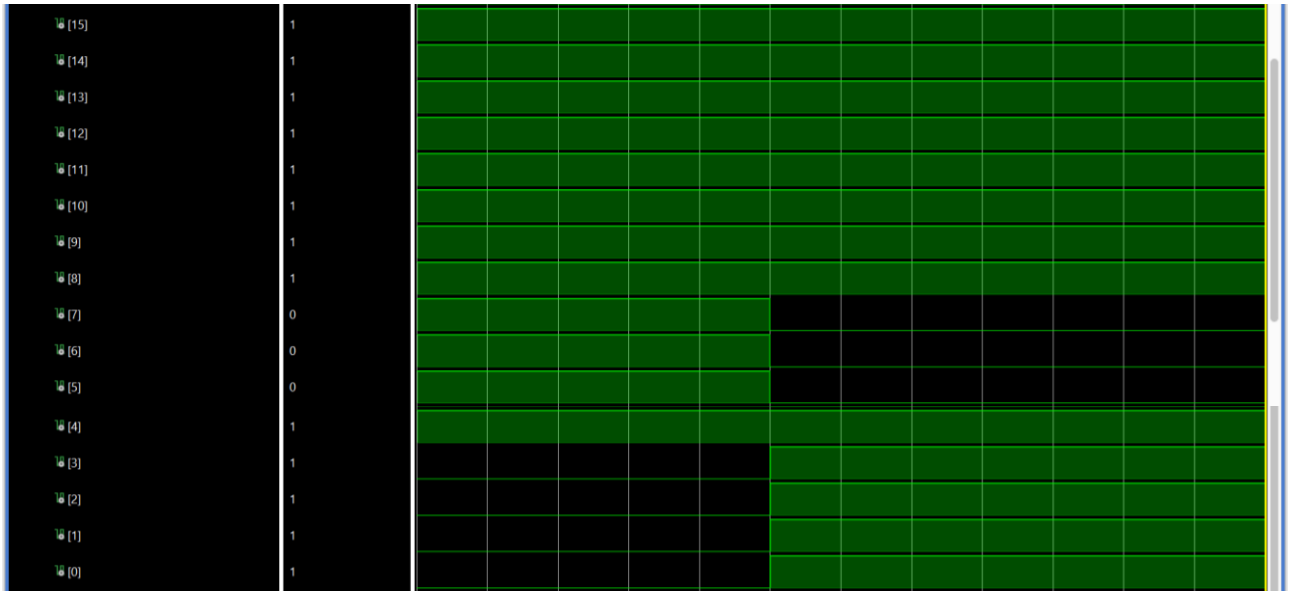
TESTBENCH



Inicialmente se observa como en función del estado de las puertas y el piso, se genera una variable que almacena esta información. Los LEDs se encienden en caso de que las puertas se encuentran abiertas y se apagan para cualquier otro caso. A continuación, se explicará la información que se ha almacenado.



Por un lado, tenemos el estado de las puertas, que corresponden a los bits 23-16. Comparándolo con los datos anteriores, vemos como, cuando las puertas se encuentran abiertas o cerradas, todos los bits de la señal se encuentran a nivel alto. Por otro lado, cuando las puertas se encuentran cerrándose, el valor almacenado es “01001000”, correspondiendo cuatro bits a un display y cuatro a otro display, y se mostrará el dibujo correspondiente a puertas cerrándose. En el caso contrario, cuando las puertas se encuentran abriéndose, el valor almacenado será “10000100” y se mostrará su correspondiente dibujo.



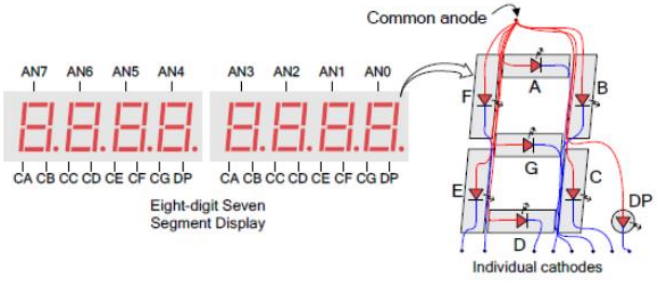
Por otra parte, los bits 15-0 corresponde a la información de cada piso. Los bits 15-12 guardan la información del piso 3, los bits 11-8 del piso 2, los bits 7-4 la del piso 1 y los bits 3-0 la del piso 0 o planta baja. Por ello, inicialmente la información guardada es “0000” en los bits correspondientes al piso 0 y, una vez cambiado de piso, la información guardada es “0001” en los bits correspondientes al piso 1.

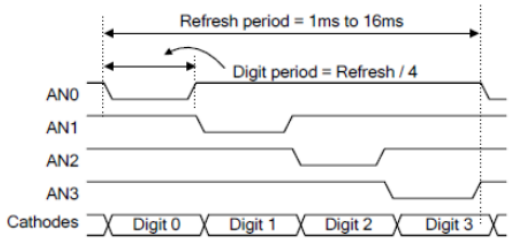
2.5.2 DISPLAYS

Como se ha mencionado anteriormente, se van a utilizar seis displays para mostrar tanto el estado del piso en el que nos encontramos como el estado de las puertas. Para ello, es necesario implementar un decodificador BCD a siete segmentos.

Los dígitos de la tarjeta son de tipo LED. Cada segmento es un LED que tiene un terminal accesible y el otro común para todos los segmentos, en nuestro caso se trata de un ánodo común, por lo que para que luzca un segmento debemos aplicar al terminal no común un '0'. Todos los dígitos comparten las mismas señales de control de los segmentos; lo que significa que todos los terminales no comunes de los segmentos A están unidos y así hasta el segmento G.

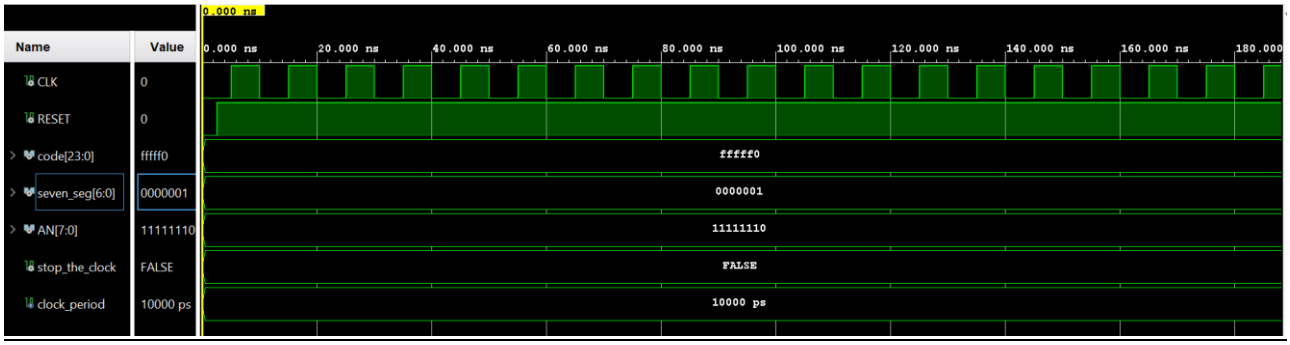
La forma de mostrar números distintos en cada dígito consiste en irlos encendiendo en secuencia haciendo que las líneas de control de los segmentos (ANx) reflejen el número correspondiente al dígito activo en ese momento. Si esta secuencia se ejecuta a suficiente velocidad, el ojo resulta engañado y percibe todos los dígitos iluminados a la vez. Para iluminar un dígito concreto en esta tarjeta es necesario aplicar un '0' a la línea de control pertinente.





				a	b	c	d	e	f	g
0	0	0	0	0	0	0	0	0	0	1
0	0	0	1	1	0	0	1	1	1	1
0	0	1	0	0	0	1	0	0	1	0
0	0	1	1	0	0	0	0	1	1	0
0	1	0	0	1	0	0	1	1	1	0
1	0	0	0	1	1	1	1	0	0	0

TESTBENCH



Para el valor almacenado en la variable *code* ("fffff0"), correspondiente con el piso 0, observamos como se activan los respectivos dígitos del vector siete segmentos ("0000001"), así como el ánodo correspondiente a dicho piso ("11111110").

3| URL GITHUB

https://github.com/lucia-phermosa/Trabajo_Sed_VHDL.git

4| BIBLIOGRAFÍA

Guiones de las prácticas de la asignatura Sistemas Electrónicos Digitales