TRABAJO
SISTEMAS ELECTRÓNICOS DIGITALES

DISEÑO DE UN ASCENSOR DE 4 PISOS EN VHDL

CURSO 24/25

Estudiantes (Matrícula)

Manuel Vallina Vides (56636)
Hugo Vigil Velasco (56648)
Daniel Vázquez Chuvieco (56638)

ÍNDICE

1. INTRODUCCIÓN y OBJETIVOS

Instrucciones para el proyecto:

Diseñar el controlador de un ascensor único en una vivienda de 4 pisos. Las entradas al circuito serán, por un lado, el piso al que el usuario desea ir mediante 4 botones, y el piso al que el ascensor está en un momento dado. Hay 2 salidas: por un lado, la del motor (2 bits) y por otro, la de la puerta (1 bit). El funcionamiento es:

- El ascensor debe ir al piso indicado por los botones, cuando llegue abrirá las puertas que permanecerán así hasta que se reciba otra llamada
- Si mientras que el ascensor se mueve, se pulsan otros botones no se debe hacer caso.
- Utilizar los LEDs y los Displays para visualizar la información.

VHDL es un lenguaje diseñado específicamente para describir tanto el comportamiento como la organización interna de sistemas digitales complejos.

En el caso del ascensor, nos permite definir no solo cómo debe responder a las solicitudes de los pisos y las señales de control, sino también cómo se conectan y funcionan entre sí los distintos componentes del sistema, como los botones, motores, sensores y pantallas de indicador.

Entre las principales ventajas de VHDL están su modularidad y capacidad de abstracción. Esto significa que podemos dividir el diseño del ascensor en partes más manejables, como el control del motor, la lógica de las llamadas, y las señales de seguridad, lo que facilita su desarrollo, prueba y mantenimiento.

Además, la simulación funcional permite verificar que el diseño cumple con los requisitos antes de implementar el hardware.

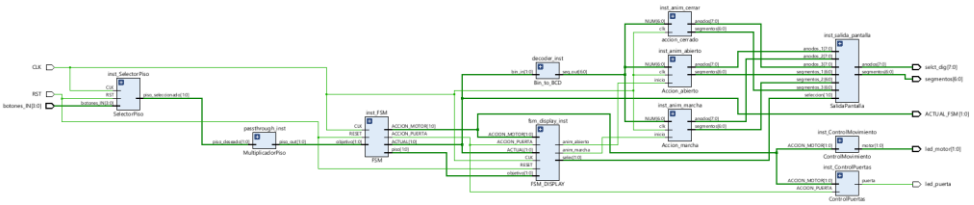
Objetivos del proyecto

Se deberá implementar mediante VHDL el control completo de un ascensor de 4 plantas con su respectiva botonera, controlando tanto la apertura y cierre de la puerta del mismo como la acción de subida o bajada llavada a cabo por su motor.

2. ENTIDADES EMPLEADAS

El diseño de un sistema de control para un ascensor en VHDL se basa en dividir el proyecto en entidades funcionales que trabajan de manera coordinada, facilitando su desarrollo y simulación. Cada entidad representa un componente esencial del sistema, conectado mediante señales bien definidas, optimizando la comunicación entre ellas.

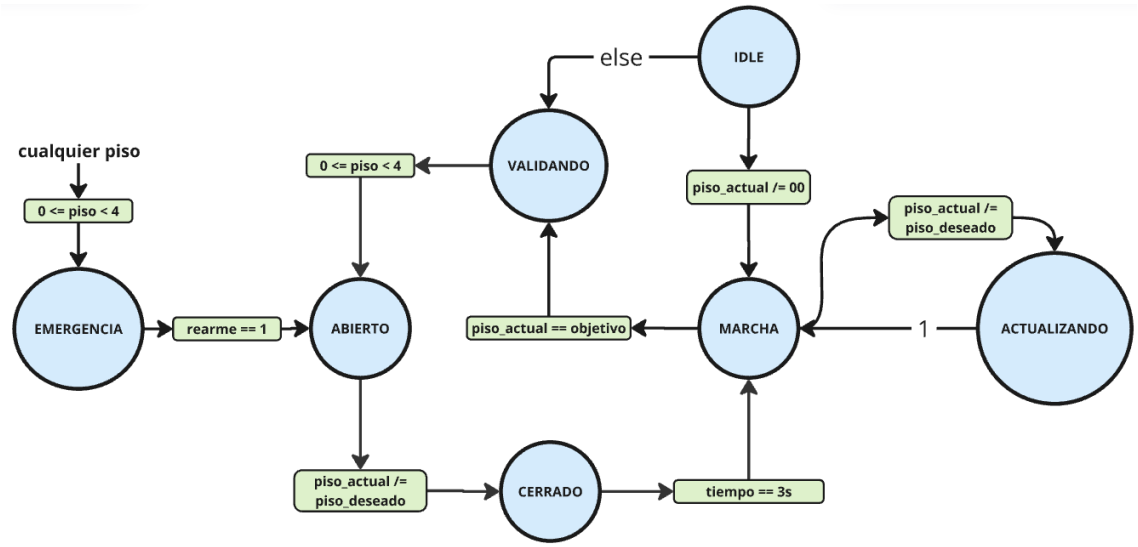
A continuación se muestran los esquemas de las distintas entidades:



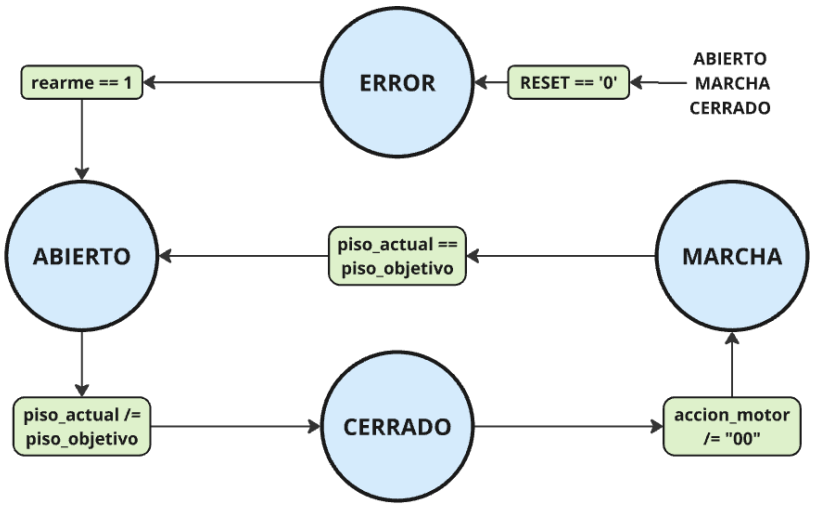
3. MÁQUINAS DE ESTADO

Este proyecto está compuesto de dos máquinas de estado: Una controla las distintas acciones del ascensor (MARCHA, REPOSO / ERROR) y de la puerta (ABRIR, CERRAR); la otra FSM se encarga únicamente de la gestión del display 7 segmentos.

- ***FSM Control Ascensor***



- ***FSM Control Display***



4. DESCRIPCIÓN DETALLADA

A nivel genérico, todas las entidades del control se encuentran instanciadas en el bloque / entidad TOP, el cual trata todas las señales de la placa y las asocia a las entradas y salidas de las demás entidades siguiendo la lógica en este proyecto establecida.

El resto de entidades tienen como función describir el funcionamiento de los distintos mecanismos del ascensor.

Se procede con la explicación de las distintas entidades y sus características:

TOP

Es la entidad principal, como ya mencionado, se encarga de gestionar y coordinar todas las demás entidades del sistema. Su propósito es integrar las funcionalidades individuales en un sistema completo y funcional.

Para ello, se definen todas las entidades necesarias, especificando claramente sus entradas y salidas. Las señales que funcionan simultáneamente como entrada y salida deben declararse como **in-out**, permitiendo su uso bidireccional.

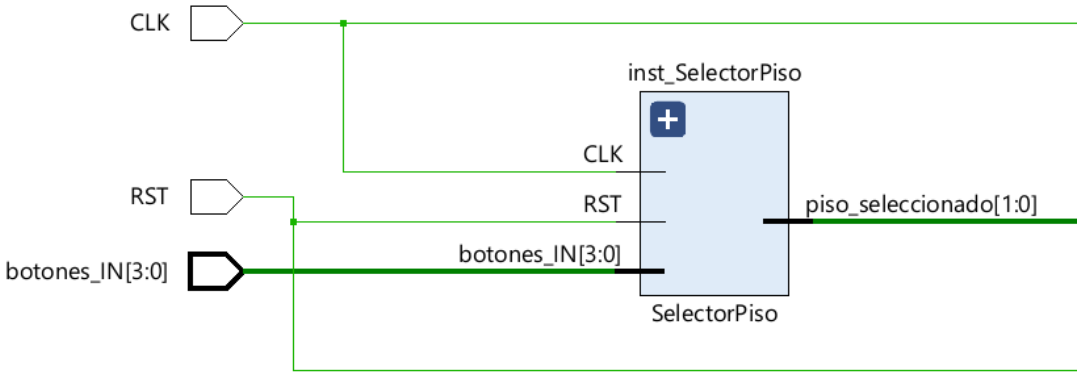
La arquitectura de este bloque se encarga de mapear las señales de cada entidad con sus valores correspondientes. Esto incluye tanto las señales físicas de la placa (como botones o sensores) como las señales internas que permiten la comunicación entre las distintas entidades del sistema.

En este diseño, se utiliza un enfoque de abstracción **estructural**, lo que significa que se hace un uso intensivo de componentes y del comando **port map**. Este último se emplea para conectar

las señales inicializadas en cada entidad con las variables reales, asegurando una interacción adecuada entre todas las partes del sistema.

Este enfoque modular facilita la organización del diseño y mejora su legibilidad, simulación y escalabilidad.

SELECTOR PISO & ANTIRREBOTES

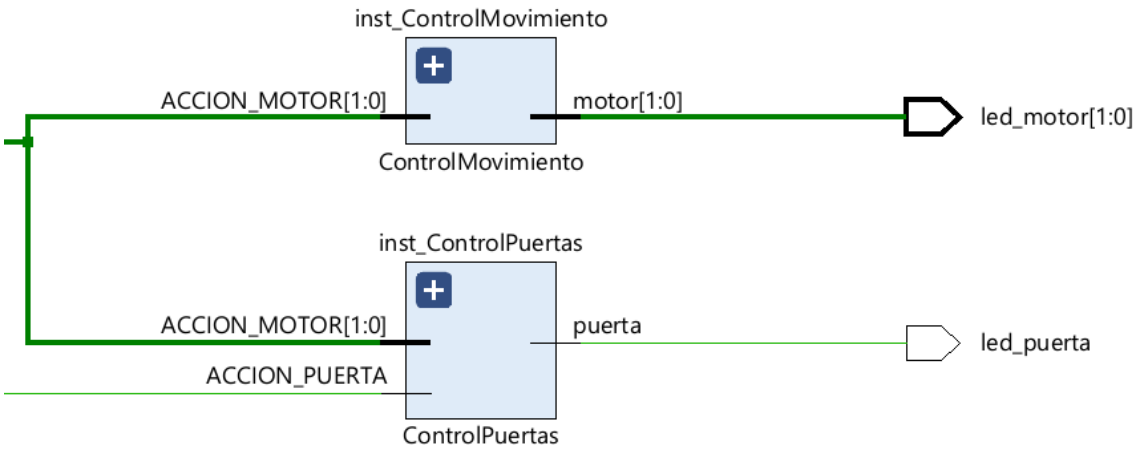


Esta entidad se encarga de la gestión del piso que marque el usuario.

Recibe la entrada de botones (*botones_IN*), la cual gracias a la entidad instanciada *botonera_anti*, filtra las señales para eliminar rebotes y selecciona el piso correspondiente.

Finalmente, devuelve el número del piso seleccionado como salida en *piso_seleccionado*.

CONTROLES MOVIMIENTO



- ***Control Movimiento***

Controla el estado del motor del ascensor de la siguiente manera:

La entrada *acción_motor* indica lo que debe hacer el motor según su valor para que suba, baje o se quede en reposo.

La salida *motor* devuelve el estado del motor del ascensor

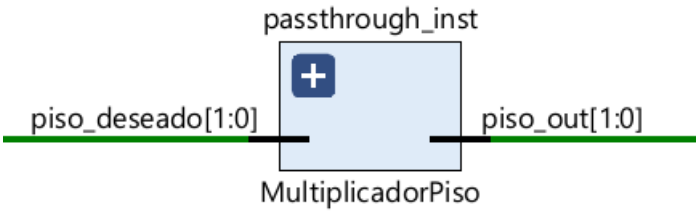
- ***Control Puertas***

Análogamente a la entidad anterior, esta gestiona el estado de las puertas del ascensor

La diferencia es que en este caso, se depende de dos entradas (*acción_puerta* y *acción_motor),* es decir, solamente se abrirán la puertas cuando el motor esté parado y se haya recibido la orden de abrir puerta.

Por último la salida *puerta* indica si esta está abierta o cerrada.

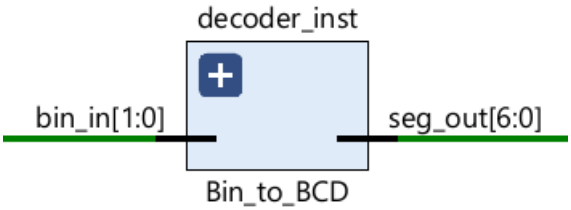
PASSTHROUGH



Esta entidad simplemente se encarga de transferir directamente el valor de un piso deseado de la entrada a la salida.

Su implementación se debe a un error que se tuvo durante el desarrollo del proyecto, que sólo con incorporar esta entidad (*MultiplicadorPiso*) se pudo solucionar.

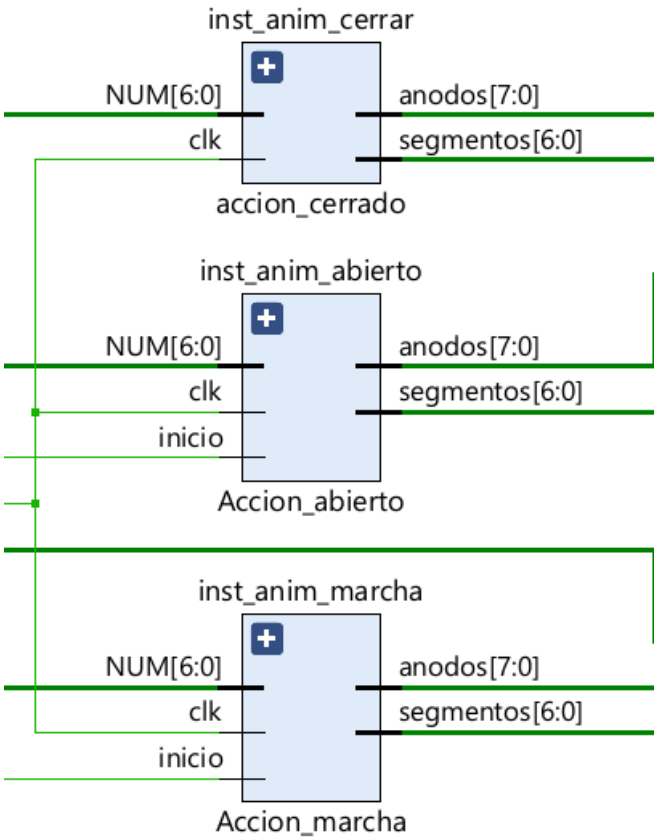
DECODER



Imprescindible para la implementación del display, traduce una señal de 2 bits (*bin_in*) a una salida compatible con este (*seg_out)*.

Permite representar el número de manera visual.

ANIMACIONES



- ***Animación Abierto***

Se encarga de controlar la animación que muestra el 7 segmentos cuando recibe la señal de *inicio* directamente desde la *FSM_display*, lo que indica que la puerta del ascensor se encuentra abierta.

Lo hace alternando entre displays activos y determinando qué segmentos se deben iluminar, mostrando números, guiones, etc.

Esto se lleva a cabo de la siguiente manera:

1. El contador *cuenta* regula la frecuencia con la que se alternan los 8 displays mediante "barridos" de tal forma que se logra el efecto a ojo humano de que los displays están encendidos constantemente para cada caso.

   Estos "barridos" son controlados por el valor de *cuenta*.

2. Cada display se activa en *selección* mediante su ánodo correspondiente

3. Con el ánodo del display correspondiente activo, se selecciona qué mostrar activando o desactivando cada segmeno según cada caso.

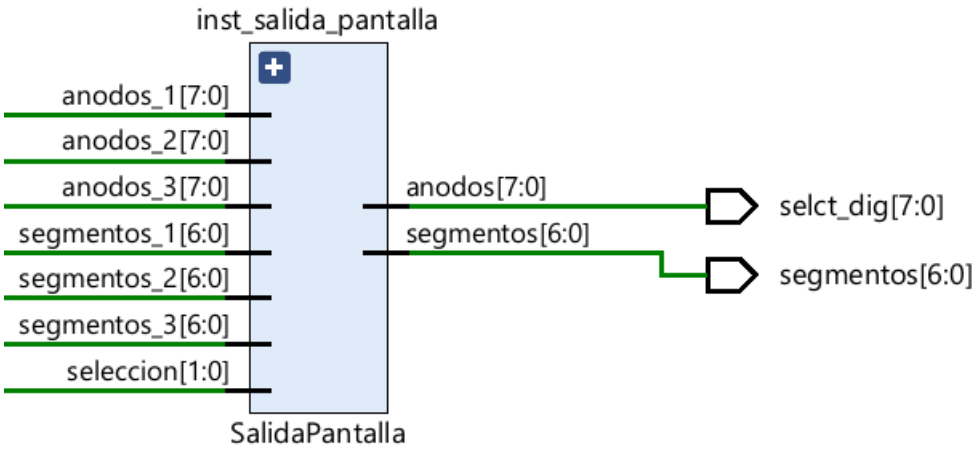
- ***Animación Marcha***

Análogo a la anterior, esta entidad gestiona, siguiendo el mismo principio de selección de ánodos, la animación que el 7 segmentos muestra cuando el ascensor está en marcha, ya sea subiendo o bajando.

- ***Animación Cerrar***

Exactamente la misma función que las dos anteriores, aplicada en este caso a la animación que se muestra cuando las puertas están cerradas.

SALIDA PANTALLA



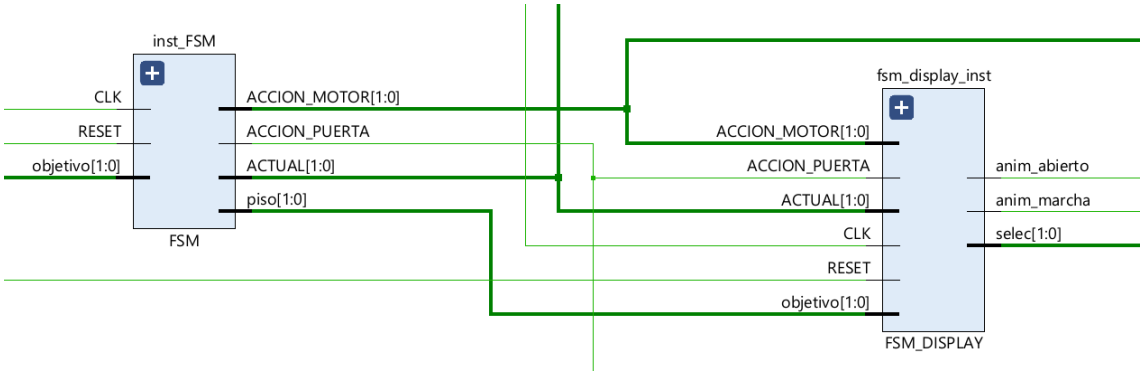
Necesaria para mostrar por pantalla las animaciones explicadas.

Recibe tres conjuntos de señales de segmentos y de ánodos de tres entidades diferentes, y mediante la señal *selección*, elige que animación muestra según uno de sus 4 valores (el valor 4 apaga todos los displays por defecto)

Es por esta entidad por lo que es posible hacer una selección dinámica de las distintas animaciones.

FSM y FSM_DISPLAY

Ya explicadas en el anterior punto.



5. PROBLEMAS GENERADOS Y SOLUCIONES

Durante el transcurso del trabajo hubo algunos problemas que conseguimos solucionar pensando entre todos y consultando diversos documentos tanto en Moodle como en internet.

- **PROBLEMA 1** → Control de la FMS del ascensor.

    - DESCRIPCIÓN DEL PROBLEMA → Tras haber pensado y tenido una FMS inicial que pensábamos que era robusta, nos sucedió un problema, durante los testbench nos dimos cuenta de que no habíamos implementado bien la actualización, pues la variable de la que dependía la condición de que se dejase de mover tenía una variable que era una entrada, también esto hacía que no se actualizase de forma correcta.

    - SOLUCIÓN → A priori el problema nos parecía más grave de lo que realmente era, nos reunimos todos juntos y poco a poco dibujamos de nuevo nuestra FMS con todos sus estados y condiciones de cambio de estado. Entonces nos dimos cuenta de que además de ese fallo, otras variables estaban mal asignadas pues ese problema podía volver a suceder, así que creamos una nueva variable que se le asignase el botón que se pulsa y esa sería la condición para que el ascensor dejase de moverse. Por otra parte, añadimos un estado más, ACTUALIZANDO, donde comprobaba el piso en el que estábamos y lo incrementaba pues antes todo esto lo hacía el estado MARCHA.

- **PROBLEMA 2** → Problemas con el 7 segmentos.

    - DESCRIPCIÓN DEL PROBLEMA → Nuestro problema consistía en que al querer hacer más llamativo y visual lo que muestra el display, queríamos que mostrase diferentes secuencias de números y figuras, pero debido a la configuración de la FPGA había que incorporar una nueva entidad, ya que es un sistema multiplexado donde todos los segmentos están conectados de forma ánodo común y muestran lo mismo.

    - SOLUCIÓN → La solución que adoptamos fue como nos adelantó el profesor, debíamos implementar la multiplexación del 7 segmentos, activando el ánodo común de un solo dígito a la vez, desactivando los otros ánodos antes de pasar al siguiente número. Esta solución no consiste en que se muestren por arte de magia los segmentos correspondientes, realmente solo es un "engaño visual", pues para que la solución sea completa hay que añadir una frecuencia lo suficiente alta para que los cambios de número no se noten a simple vista, sino se verían como parpadeos.

- **VARIOS PEQUEÑOS PROBLEMAS** → A lo largo del transcurso del trabajo surgieron pequeños problemas que conseguimos solucionar, pero nos hicieron perder tiempo.

    - EJEMPLO 1 → Al usar la FPGA por primera vez, no sabíamos que tenía el reset negado, por lo que nuestro código solo funcionaba si pulsábamos el reset, la solución fue tan simple como cambiar el valor del reset.

    - EJEMPLO 2 → Un problema que nos ha sucedido es que, al tener tantos códigos, a veces no sabíamos en que actualización estábamos, y algunas declaraciones de variables en la zona de IP.

    - EJEMPLO 3 → A veces la generación de testbench no era adecuada y no simulaba bien todos los casos, y lo solucionamos haciendo que ChatGPT nos hiciese el testbench con las especificaciones que le pedíamos.