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

Trabajos Sistemas Electrónicos Digitales

CURSO 2023/24

Título: Diseño e implementación en VHDL de un ascensor de 4 pisos

FECHA: 20/06/2024

Miembros:

Sonia Carrero Díaz (nº matrícula: 55784)
Iker Castiella Aguirrezabala (nº matrícula: 55791)

ÍNDICE:

1- INTRODUCCIÓN:

VHDL (VHSIC Hardware Description Language) es un lenguaje computacional empleado en circuitos digitales que fue creado en los años 80 por el Departamento de Defensa de los Estados Unidos de América.

El lenguaje permite tanto la descripción del comportamiento de los circuitos como su estructura interna. Esto incluye especificar cómo deben funcionar los circuitos y cómo se interconectan sus componentes. VHDL también facilita la simulación y prueba de diseños antes de fabricarlos, ayudando a detectar y corregir errores de manera más sencilla. Además, los diseños en VHDL pueden ser convertidos en hardware real, como FPGA o ASIC.

Un programa en VHDL contiene librerías y paquetes con definiciones y funciones, una entidad que define las entradas y salidas del circuito, y una arquitectura que describe la implementación interna del circuito. Las ventajas de VHDL incluyen la capacidad de abstracción y modularidad en el diseño, facilitando la simulación y verificación funcional.

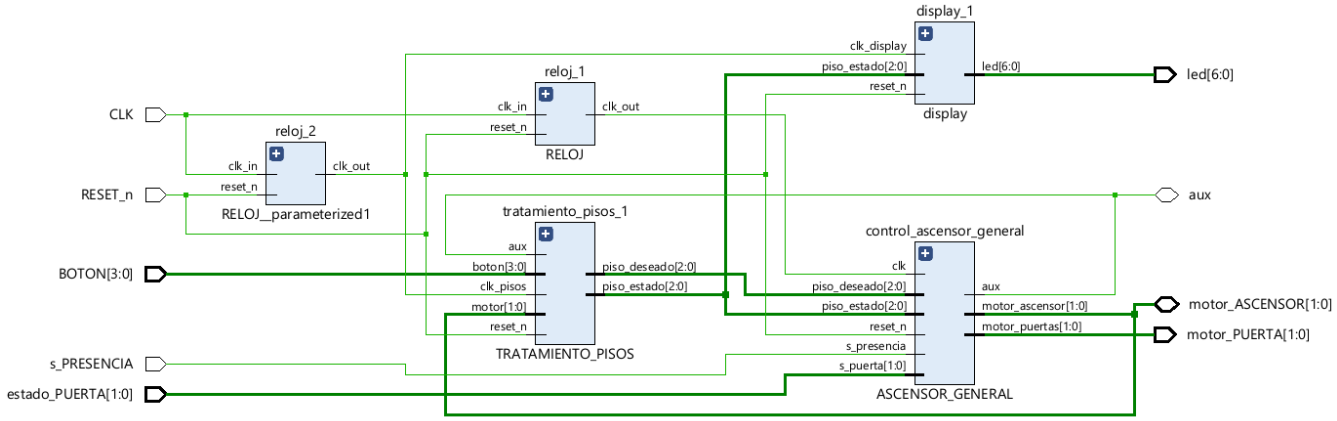
2- OBJETIVO:

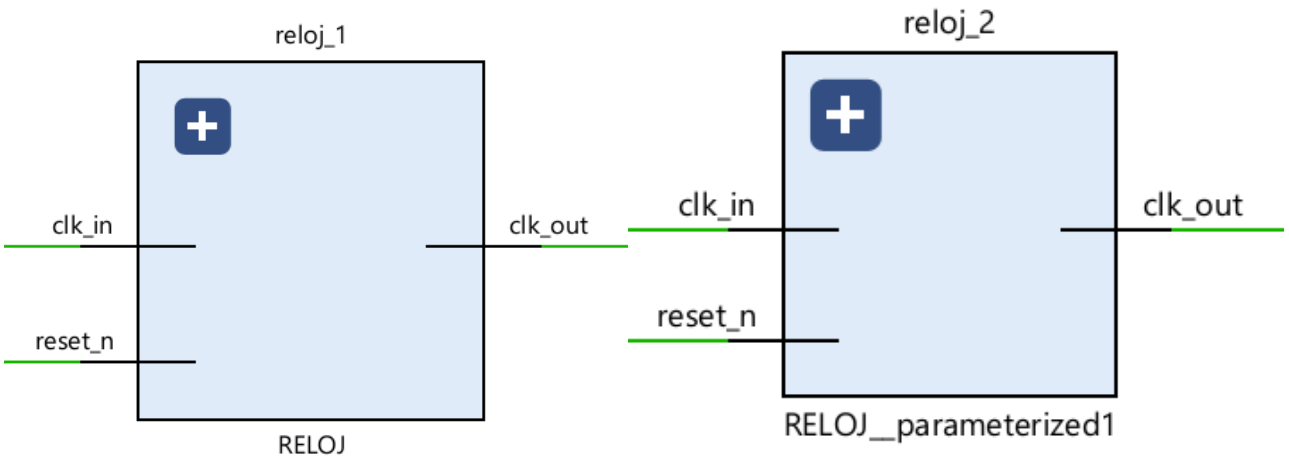
El objetivo del proyecto consiste en implementar mediante la descripción hardware en VHDL el funcionamiento y control completos de un ascensor. El ascensor en cuestión se diseñará para un hipotético edificio con 4 pisos, con su respectiva botonera, motor de traslación vertical y casuística de las puertas de acceso.

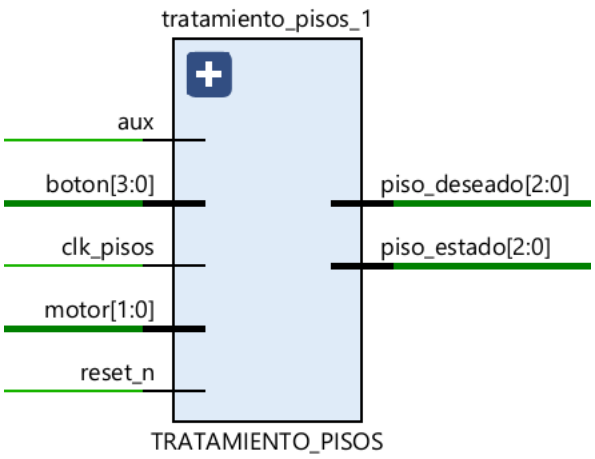
Para ello se hará uso de la documentación otorgada por los profesores de la asignatura a lo largo del curso académico.

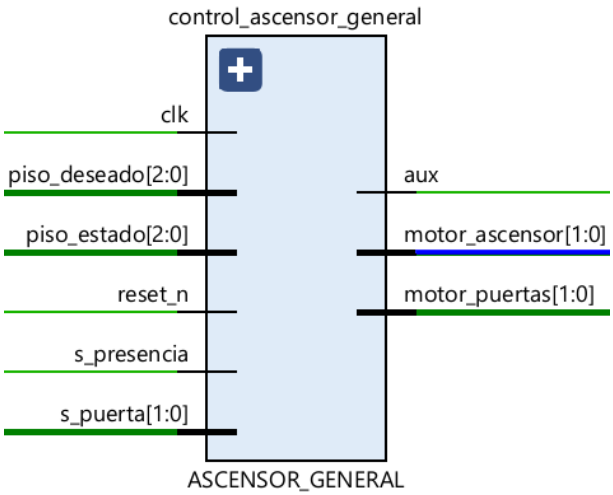
3- DIAGRAMAS DE BLOQUES/ENTIDADES:

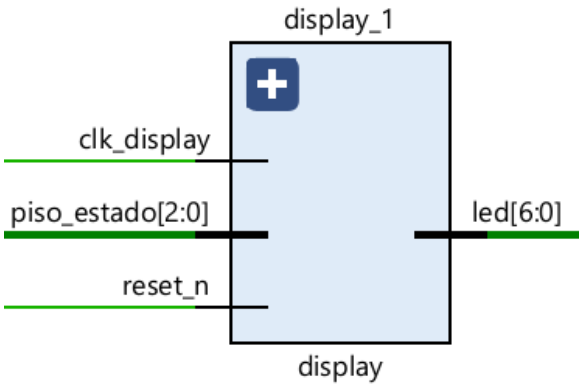
Los diagramas de bloques muestran las conexiones entre señales de las distintas entidades que describen el funcionamiento de nuestro proyecto. En la parte izquierda de cada entidad se sitúan las señales de entrada, mientras que las señales situadas a la derecha son las de salida. Como se puede apreciar en el diagrama general, las señales reales de entrada y salida son aquellas que se declaran en la entidad principal de control (TOP), mientras que el resto son señales internas de cada entidad que sirven para favorecer una comunicación satisfactoria. Existen diferentes señales que se emplean como entradas y salidas simultáneamente para diferentes entidades, luego éstas serán declaradas como inout's a la hora de realizar el código.






tratamiento_pisos_1
aux
boton[3:0]
clk_pisos
motor[1:0]
reset_n
piso_deseado[2:0]
piso_estado[2:0]
TRATAMIENTO_PISOS


control_ascensor_general
clk
piso_deseado[2:0]
piso_estado[2:0]
reset_n
s_presencia
s_puerta[1:0]
aux
motor_ascensor[1:0]
motor_puertas[1:0]
ASCENSOR_GENERAL


display_1
clk_display
piso_estado[2:0]
reset_n
led[6:0]
display

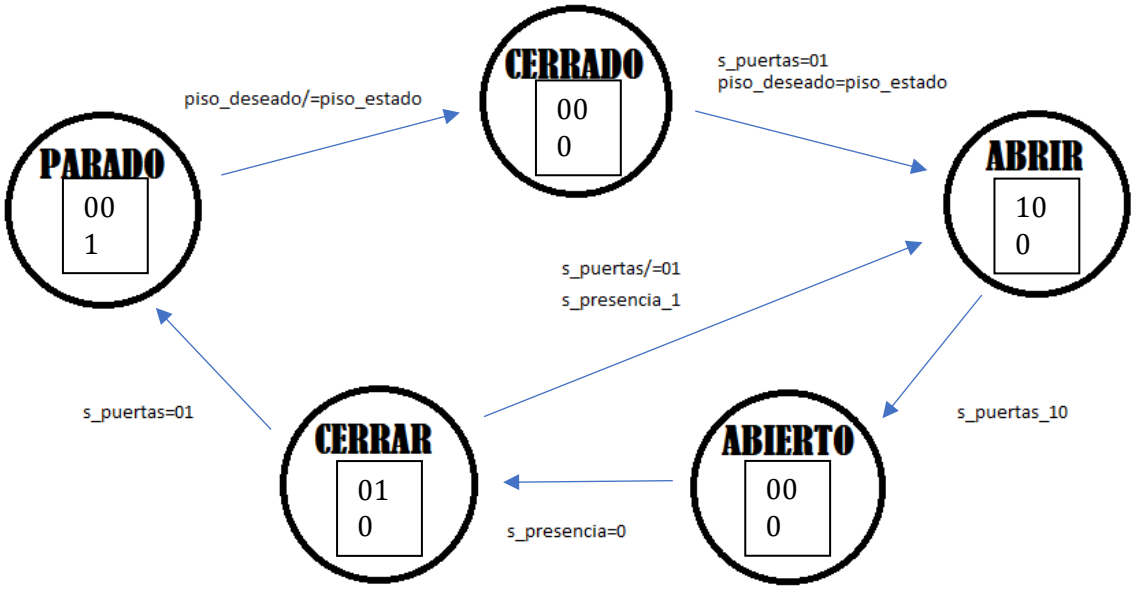
4- MAQUINAS DE ESTADO:

Se han diseñado dos máquinas de estado para la entidad de ASCENSOR_GENERAL. Estos diagramas representan el funcionamiento de los process incluidos en ese bloque. El primero controla el funcionamiento de las puertas y el otro el funcionamiento del motor del ascensor. A continuación, se indicará las entradas y salidas empleadas en ambos juntos a sus respectivos diagramas.

- **CONTROL DE LAS PUERTAS:**

  Entradas: s_presencia, s_puertas, piso_deseado, piso_estado

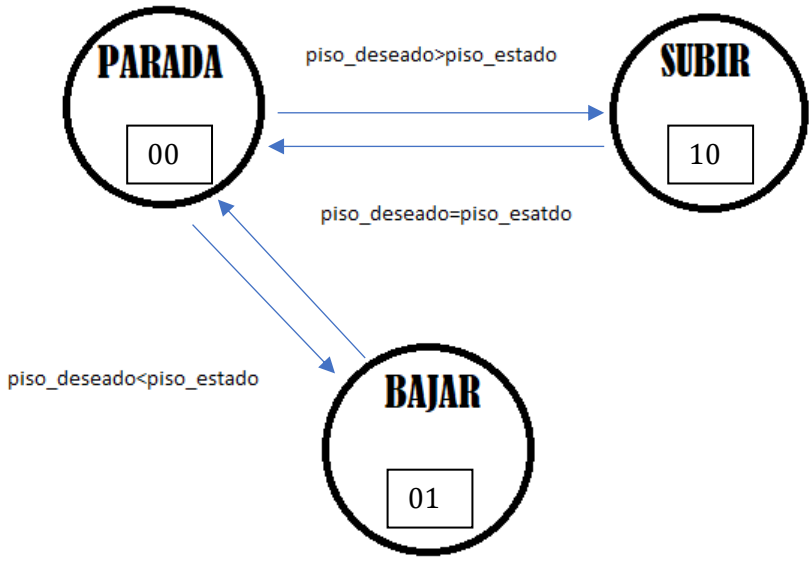
  Salidas: aux (0,1) , motor_puertas (00, 10, 01)



- **CONTROL DEL MOTOR DEL ASCENSOR:**

  Entradas: piso_deseado, piso_estado

  Salidas: motor_ascensor (00, 10, 01)



5- DESCRIPCIÓN DEL CONTROL COMPLETO DEL ASCENSOR:

Como se menciona anteriormente, el código está divido en diferente bloques o entidades para tener un mejor funcionamiento y organización. El bloque principal TOP se encarga de tratar las señales reales de la placa y las asocia a las entradas y salidas de las demás entidades siguiendo un orden y lógica determinados. Posteriormente, las entidades secundarias son las encargadas de describir el control y funcionamiento de distintos mecanismos del ascensor. A continuación, se hará una breve descripción de cada entidad y se explicará el funcionamiento y características que posee.

- TOP:

Es el bloque principal y general, y tiene como funcionamiento coordinar todas las entidades. Primero, se define cada entidad, indicando cuáles serán las entradas y salidas de cada una de ellas. Aquellas variables que se emplean en forma de entrada y salida deben inicializarse como inout`s.

También se ha de diseñar la arquitectura del bloque. En ella se asocian las variables inicializadas de cada entidad con los valores reales: tanto las señales reales de la placa, como las señales intermedias que actúan como entradas y salidas.

En este caso el nivel de abstracción empleado es el Structural, pues se hace reiteradamente uso del Component y de la función portMap (asignación de variables).

- RELOJ:

La funcionalidad del bloque es modificar la frecuencia del reloj de la placa al requerido por nuestro circuito. Para ello se crearán dos relojes a frecuencias diferentes, uno a 100MHz y otro a 150MHz. En este caso se escribe el código con el nivel de BEHAVIORAL.

- DISPLAY:

Esta entidad se encarga del control de la pantalla de la placa, principalmente para que indique el piso en el que se encuentra el ascensor. Las variables empleadas son: el reloj, el reset y la variable que indica el piso actual en el que se encuentra el ascensor. La arquitectura empleada es tipo BEHAVIORAL.

- TRATAMIENTO_PISOS:

Este bloque tiene como función, como bien dice el título, controlar el tratamiento de los pisos, y servirá como motor principal para el control del ascensor en el resto de las entidades. Esta entidad recibe (además de la señal de reloj y el reset) la botonera donde se selecciona el piso al que se quiere ir, el motor principal, y una señal auxiliar (aux). El motor principal como entrada sirve para modificar el estado en el que se encuentra el ascensor. Si el ascensor sube, el piso aumenta, y viceversa. La señal aux sirve para habilitar o deshabilitar la botonera en función de si el ascensor esta en movimiento o parado.

Las variables de salida son el estado actual del piso y el estado deseado del piso. Serán empleadas en el display y en el control general del ascensor.

Un process trata el piso deseado en función del botón que se selecciona. Para ello se realiza un case con las 5 opciones que existen: piso 1, piso 2, piso 3, piso 4 y piso en el que se encontraba. En caso de reset se selecciona el primer piso.

También se hace un segundo process que trata el estado del piso en función de la señal del motor. S el ascensor sube (motor=10) se incrementa el piso, y si baja (motor=01) disminuye.

Para este bloque se hace el diseño nuevamente en BEHAVIORAL.

- ASCENSOR_GENERAL:

Este bloque se encarga del funcionamiento del motor de las puertas y del motor general del ascensor. Para ello tiene como variables de entrada (además del reset y el reloj) el estado del piso, el piso deseado, un sensor de presencia y un sensor de la puerta.

Existen dos process principales. Uno encargado del motor general y otro del motor de las puertas. Para la simplificación y para una mejor organización de código se han creado unas variables de TIPO para indicar el estado en el que se encuentren ambos motores.

El process del motor general funciona de la siguiente manera. Sus variables de tipo TYPE son ACTUAL_MOTOR y SIGUIENTE_MOTOR, que valen STANDBY, SUBIR o BAJAR. Si se encuentra en STANDBY, el motor permanecerá parado, pero si el piso deseado no es el piso estado, se activará el motor para subir o bajar y la variable SIGUIENTE_MOTOR será SUBIR o BAJAR. En caso de que el ascensor esté subiendo o bajando, el momento en el que el piso estado coincida con el deseado, se parará el motor y volverá al STANDBY.

El process del motor de las puertas tiene una logística algo más complicada. En este caso, las variables de tipo TYPE son ACTUAL_PUERTAS y SIGUIENTE_PUERTAS, que valen ABRIR, CERRAR, ABIERTO, CERRADO o PARADO. Si se encuentra en CERRADO, el motor de las puertas permanece inactivo; sucede cuando el ascensor está en movimiento. Si la puerta se encuentra cerrada y el piso deseado coincide con el estado, pasa a ABRIR, si no, permanece en CERRADO. Si se encuentra en ABRIR, el motor de las puertas se activa (10=abriendo) y cuando el sensor de las puertas detecte que ya se han abierto se pasa al estado de ABIERTO y se apagan los motores. Cuando se encuentra en ABIERTO y el sensor de presencia está inactivo, pasa a CERRAR, si no, permanece en ABIERTO. Si está en CERRAR, el motor se vuelve a activar (01=cerrando). Si el sensor de presencia se activa, pasa nuevamente al estado ABRIR, y si no se detecta presencia y el sensor de las puertas detecta que están cerradas, pasa al estado PARADO y se desactivan los motores. La diferencia entre el estado de PARADO y CERRADO es que en uno se habilita la botonera y en el otro se bloquea. Cuando las puertas se cierran y pasa al estado PARADO, la variable aux pasa a valer 1 (habilitando la botonera) y queda a la espera de que se seleccione un piso diferente al piso en el que se encuentre el ascensor. Una vez sucede, pasa al estado CERRADO.

Por último, existen otros dos process más cuya función es actualizar los estados de cada motor. Las variables SIGUIENTE_PUERTAS y SIGUIENTE_MOTOR se pasan a ACTUAL_PUERTAS y ACTUAL_MOTOR respectivamente con cada pulso de reloj.

Para esta entidad se hace uso nuevamente de la arquitectura BEHAVIORAL.

6- PROBLEMAS GENERADOS Y SOLUCIONES PROPUESTAS:

Problema 1:

Inicialmente, habíamos separado el control de las puertas y el motor general en entidades distintas con el objetivo de modularizar y organizar mejor el diseño. Sin embargo, durante la implementación, nos dimos cuenta de que había una dependencia significativa entre ambos componentes.

Alternativa 1:

Para mejorar la coherencia y la eficiencia del sistema, decidimos integrar el control de puertas y el motor general en una única entidad. De esta manera, se logró una mejor coordinación y sincronización de las acciones, reduciendo la complejidad del diseño y facilitando el mantenimiento futuro, al tener todos los componentes relacionados en un solo lugar.

Problema 2:

Durante las pruebas iniciales del sistema, encontramos que el funcionamiento de la botonera no era adecuado. Observamos, que, sin la presencia de una variable de control, el sistema respondía de manera impredecible a las señales de entrada. Esto se debía a que la botonera podría activarse en momentos no deseados, como durante el movimiento del ascensor o cuando las puertas aún estaban en proceso de apertura o cierre.

Alternativa 2:

Introdujimos la variable “aux” para controlar la habilitación de la botonera. Esta variable permite autorizar la funcionalidad de la botonera solo cuando es seguro y apropiado hacerlo, como cuando el ascensor está en estado “PARADO”.

Problema 3:

Inicialmente, utilizamos frecuencias de reloj diferentes para diversos procesos dentro del ascensor. Esto condujo a problemas de coordinación y sincronización entre los distintos componentes del sistema, especialmente entre el control de puertas, el motor general y la actualización del estado del ascensor (display incluido). La falta de una frecuencia de reloj uniforme provocó que algunos eventos no se ejecutaran en el orden esperado o no se completaran de manera oportuna. Por ejemplo, el cierre de las puertas podría no iniciar justo después de que el ascensor llegara a un piso deseado, lo que afectaba la eficiencia y la seguridad del ascensor.

Alternativa 3:

Para abordar este problema, realizamos pruebas directamente con la placa de desarrollo. Ajustamos las frecuencias de reloj para garantizar una coordinación óptima entre todos los procesos del ascensor. Establecer valores adecuados de frecuencia nos permitió sincronizar perfectamente las operaciones de apertura/cierre de puertas, el movimiento del motor general y la actualización del estado del ascensor, asegurando así un funcionamiento eficiente del sistema en todas las condiciones.

7- ENLACES:

LINK DE ACCESO AL REPOSITORIO GITHUB (VIDEO DEMOSTRATIVO INCLUIDO)

- Soniacarrero/VHDL (github.com)