CONTROL DE UN ASCENSOR DE 4 PISOS

Trabajo VHDL Sistemas Electrónicos Digitales (2022-2023)

Ingeniería Electrónica Industrial y Automática

Lucía Antolín Pulido

Nº de matrícula: 55125

Eva Gómez Barahona

Nº de matrícula: 55267

Marta Seisdedos Barrientos

Nº de matrícula: 55468

Índice

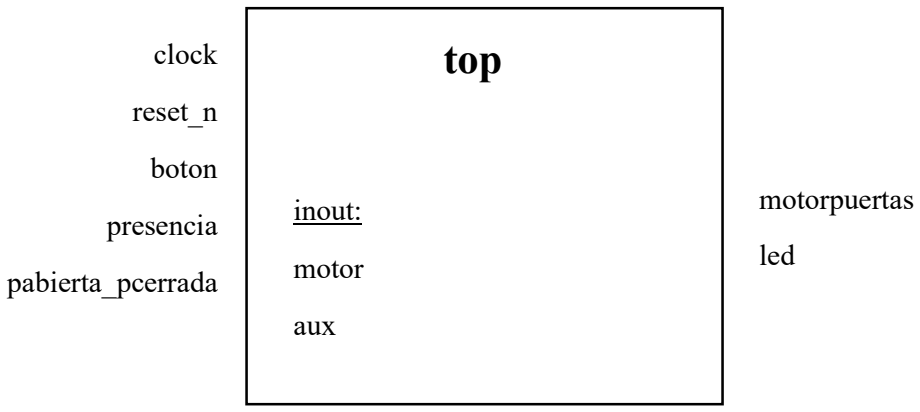
INTRODUCCIÓN

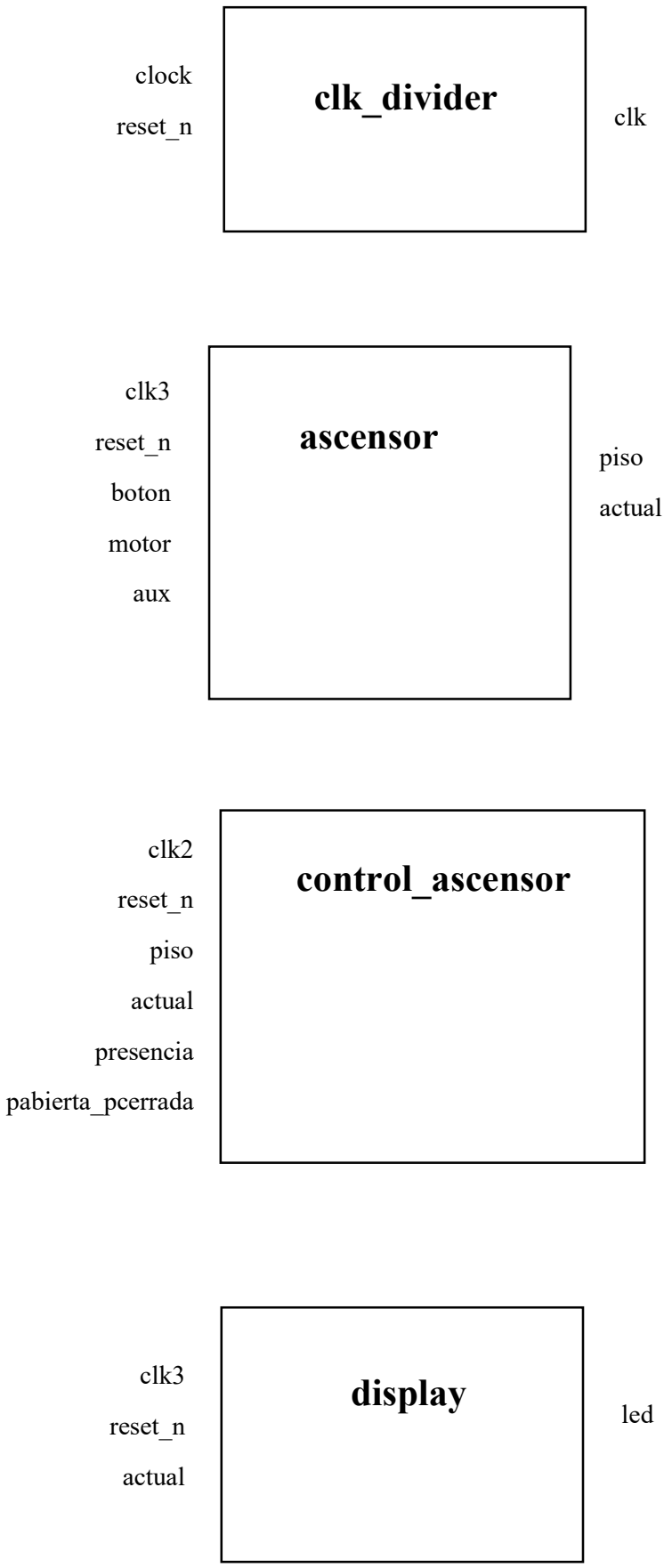
Este proyecto se ha realizado de acuerdo con la documentación que la asignatura nos ha aportado a lo largo del curso. En un primer lugar, vamos a llevar a cabo el desarrollo del funcionamiento de un ascensor mediante la descripción hardware en VHDL.

El diseño que se ha llevado a efecto es un ascensor de cuatro pisos con un funcionamiento diferenciado en dos bloques, cada uno con sus respectivos componentes. Por una parte, tratamos la cabina, es decir, la botonera. Y, por otra parte, tenemos el control, el cual se basa en la apertura de puertas y el movimiento del motor del ascensor.

DIAGRAMAS

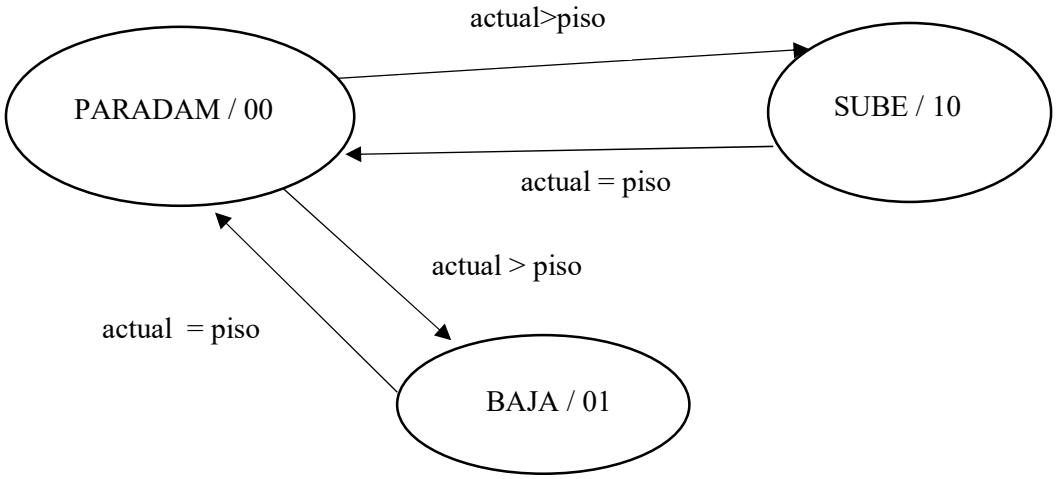
En primer lugar, se muestra de manera esquemática la descripción de cada bloque empleado, con sus respectivas entradas (situadas a la izquierda) y salidas (situadas a la derecha). Como se va a describir con posterioridad más detalladamente, el bloque "top" engloba al resto de bloques que interactúan entre ellos dentro de él. Por este motivo, las entradas y salidas de la placa (es decir, las globales del proyecto) son las del bloque "top", y las demás serán variables internas de dicho proyecto.



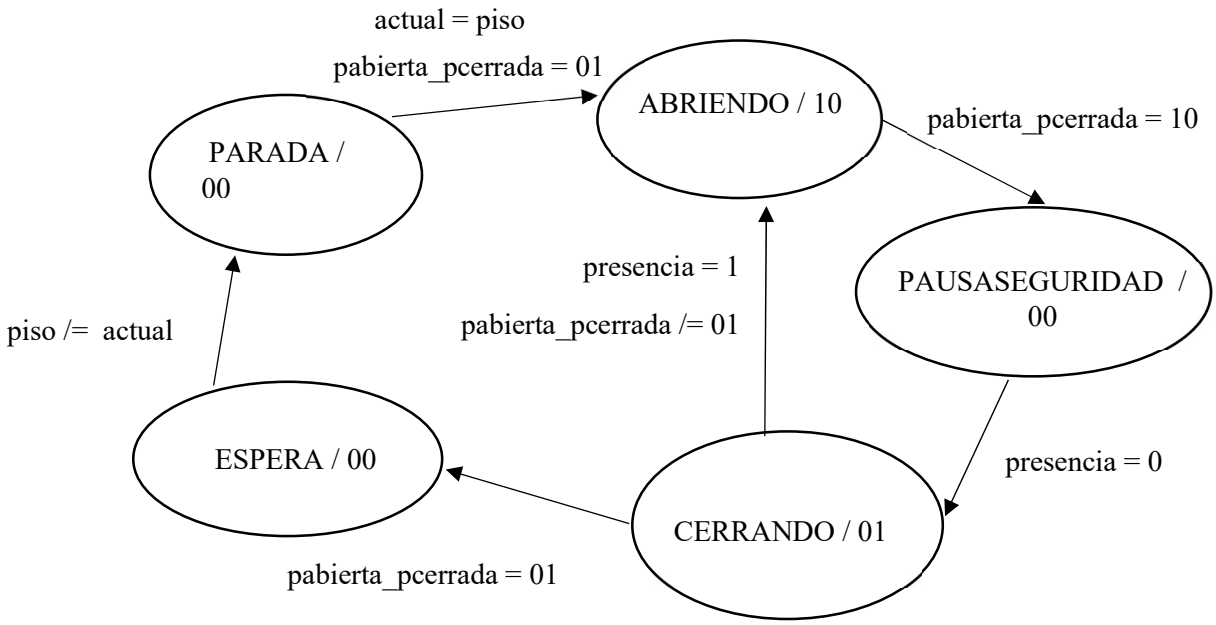

clock
reset_n
clk_divider
clk
clk3
reset_n
boton
motor
aux
ascensor
piso
actual
clk2
reset_n
piso
actual
presencia
pabierta_pcerrada
control_ascensor
aux
motor
motorpuertas
clk3
reset_n
actual
display
led

A continuación, se describen las máquinas de estados de Moore desarrolladas en el bloque “control_ascensor”, una para los estados por los que pasa el motor del ascensor, y otra para los estados de las puertas del ascensor.

- Máquina de Moore para el motor del ascensor.
  Entradas: actual (piso en el que me encuentro), piso (piso al que deseo ir).
  Salida: motor (motor del ascensor que indica si está subiendo (motor=10), bajando (motor=01) o parado (motor=00)).



- Máquina de Moore para las puertas del ascensor.
  Entradas: pabierta_pcerrada, presencia, actual, piso.
  Salida: motorpuertas (motor de las puertas del ascensor que indica si están abriéndose (motorpuertas=10), cerrándose (motorpuertas=01) o paradas (motorpuertas=00)).



FUNCIONAMIENTO: FUNCIONES DESARROLLADAS E INTERFAZ DEL PROGRAMA

Como se ha indicado anteriormente, para facilitar la programación y el funcionamiento del ascensor se ha propuesto una arquitectura definida por diferentes bloques. Estos se encargan de una función de forma individual y finalmente son unificados en el bloque top.
El proyecto está propuesto por los siguientes bloques:

ASCENSOR:

En este primer bloque se hace referencia a lo que sería la maqueta física del ascensor. Es por ello por lo que se recibe como entrada los actuadores que pulsa el usuario: botones de dentro o fuera (*boton*), además del RESET (r*eset_n*) . Por otro lado, como salida se obtendrá el piso actual (*actual*) y el piso deseado (*piso*). Estas dos salidas serán usadas en el bloque de control para permitir enviar el movimiento correspondiente. Además, se ha añadido una entrada (*aux)* que inhabilita la pulsación de los botones cuando el ascensor está en marcha. Esto garantiza el correcto funcionamiento del dispositivo.

Se han creado dos process los cuales se encargarán de garantizar el correcto movimiento. La asignación del piso correspondiente a cada botón se realiza en *piso_deseado* a partir de un case (simplificando así la complejidad de la programación). Además, aquí se establece el piso por defecto en caso de que se produzca un reset en el dispositivo (piso

1). En el siguiente process denominado *movimiento* se asigna al ascensor el piso actual en el que está en función del movimiento de este (a partir de la variable motor que puede tomar los valores: 01 si está bajando o, 10 si está subiendo).

CONTROL ASCENSOR:

Por otro lado, este bloque es el encargado de mandar las órdenes para subir o bajar, abrir o cerrar las puertas etc… Siempre en función de lo que haya recibido como entrada. Es por ello por lo que en este bloque es necesario tener como variables de entrada los sensores utilizados en el dispositivo. Estos serán los encargados de detectar la presencia de un usuario (*presencia*) y garantizar que las puertas se han abierto o cerrado correctamente (*pabierta_pcerrada*). Además, este bloque necesita saber la información del bloque *ASCENSOR*, por lo que las variables encargadas de detectar el piso en el que está actualmente y al que quiere ir se implementarán como entradas. En función de estas variables se procederá a seguir la máquina de estados detallada anteriormente en el process del bloque (denominado *maquina*). Además, se han establecido *clocks* los cuales se encargan de controlar el tiempo que se mantienen las puertas abiertas para permitir el paso de los usuarios al interior o exterior de la cabina y, por otro lado, establecer una velocidad de movimiento del ascensor. Para controlar el cambio de estados en los diagramas diseñados y garantizar un buen funcionamiento del reset se han creado los process *clockmotor* y *clockpuertas.*

CLK_DIVIDER:

Este bloque se encarga de crear señales de reloj a distintas frecuencias. Esto posibilita el funcionamiento correcto de todos los relojes implementados en el dispositivo.

DISPLAY:

Este bloque tiene como entradas las variables que establecen el piso actual donde se encuentra el ascensor (*piso_actual*), el botón de reset (*reset_n*) y el reloj encargado en establecer el movimiento del ascensor (*clk3*). En el process desarrollado en el bloque (*pdisplay*) se asigna cada piso al número que se quiere visualizar en el display.

TOP:

Finalmente, todos los bloques creados anteriormente son unificados en el bloque top.vhd para permitir un funcionamiento total y sincronizado. Para ello lo primero que se hace es establecer la entidad de este, nombrando las entradas y salidas globales del proyecto. Además, las variables que se utilizan como entrada o salida dependiendo del bloque en el que se utilicen serán de tipo inout para garantizar así el buen funcionamiento.

A continuación, se desarrolla la arquitectura de este, definiendo las señales necesarias (que serán posteriormente asignadas en los bloques necesarios). Por otro lado, se establecerán todos los componentes provenientes de los subbloques desarrollados anteriormente. Aquí se procederá a la asignación de las variables.

PROBLEMAS OCASIONADOS Y SOLUCIONES ADAPTADAS

Durante el desarrollo del programa nos han surgido una serie de problemas. En un principio, comenzamos creando una máquina de estados que comprendía toda la mecánica del ascensor, sin hacer ninguna distinción. El fallo de este planteamiento era que, al haber determinado una variable para el botón para fuera del ascensor y otra para los botones de dentro, el hecho de almacenar el valor del piso al que se dirige el ascensor después de pulsar el botón deseado nos resultaba complejo.

Mientras buscábamos una solución, añadimos al proyecto un archivo con el control del motor del ascensor, otro para el control del motor de las puertas, otro para valorar el piso en el que estamos, y dependiendo el motor sube, baja o se para, y finalmente, otro para la botonera.

Desde un principio, planteamos un diseño con muchos archivos, por lo que decidimos sintetizar el código. De este modo, veíamos mucho más práctico el programa juntando el control del motor y el de las puertas con la máquina de estados inicial.

La última modificación a nuestro diseño del proyecto fue separar a la cabina del ascensor, por un lado, y, por el otro, el control del ascensor, debido a que el esquema del programa tenía más sentido de esta forma. De todas formas, nos empezaron a aparecer varios errores a la hora de cargar el programa en la placa *Nexys 4 DDR*. Por ejemplo, no nos funcionaba la apertura de puertas, es decir, no se iluminaban los leds cuando el ascensor llegaba al piso.

Con el objetivo de arreglar dicho error, creamos una variable auxiliar para que esta se encuentre desactivada durante el movimiento del ascensor, y solamente cuando el ascensor llegue al piso deseado, esta variable se active. De esta manera, ya nos funcionaba.

Como solución al problema del almacenamiento del valor del piso, dentro de la botonera, decidimos reducir el número de botones a uno, el cual funciona tanto para pulsar desde dentro como desde fuera, y determinado que en el caso de que no se pulse ningún botón se asigna la variable que guarda el piso actual a la variable del piso al que quiero ir.

Por último, tuvimos algún problema con las frecuencias de los distintos relojes a la hora de probar el programa en la placa, ya que habíamos asignado a cada uno tenía una frecuencia distinta previamente. No obstante, lo solucionamos ajustando las frecuencias de los relojes acorde con las acciones a las que se les había asignado a cada uno de ellos.

LINK REPOSITORIO GITHUB

https://github.com/evagombar/Trabajo-SED-VHDL-FPGAs